



**Thank You for your purchase
Linux Foundation CKS Exam Question & Answers
Certified Kubernetes Security Specialist Exam**

Product Questions: 48

Version: 6.0

Question: 1

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.

**Answer: See the
Explanation below:**

Explanation:

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/<podname> -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: 2

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:-

- a. Ensure the --authorization-mode argument includes RBAC
- b. Ensure the --authorization-mode argument includes Node
- c. Ensure that the --profiling argument is set to false

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

- a. Ensure the --anonymous-auth argument is set to false.
- b. Ensure that the --authorization-mode argument is set to Webhook.

Fix all of the following violations that were found against the ETCD:-

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

Hint: Take the use of Tool Kube-Bench

Answer: See the Explanation below.

Explanation:

API server:

Ensure the --authorization-mode argument includes RBAC

Turn on Role Based Access Control.

Role Based Access Control (RBAC) allows fine-grained control over the operations that different entities can perform on different objects in the cluster. It is recommended to use the RBAC authorization mode.

Fix - Buildtime

Kubernetes

apiVersion: v1

kind: Pod

metadata:

creationTimestamp: null

labels:

component: kube-apiserver

```
tier: control-plane
name: kube-apiserver
namespace: kube-system
spec:
  containers:
    - command:
      + - kube-apiserver
      + --authorization-mode=RBAC,Node
        image: gcr.io/google_containers/kube-apiserver-amd64:v1.6.0
        livenessProbe:
          failureThreshold: 8
          httpGet:
            host: 127.0.0.1
            path: /healthz
            port: 6443
            scheme: HTTPS
        initialDelaySeconds: 15
        timeoutSeconds: 15
      name: kube-apiserver-should-pass
      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
```

Ensure the --authorization-mode argument includes Node

Remediation: Edit the API server pod specification file /etc/kubernetes/manifests/kube-apiserver.yaml on the master node and set the --authorization-mode parameter to a value that includes Node.

--authorization-mode=Node,RBAC

Audit:

```
/bin/ps -ef | grep kube-apiserver | grep -v grep
```

Expected result:

'Node,RBAC' has 'Node'

Ensure that the --profiling argument is set to false

Remediation: Edit the API server pod specification file /etc/kubernetes/manifests/kube-apiserver.yaml on the master node and set the below parameter.

```
--profiling=false
```

Audit:

```
/bin/ps -ef | grep kube-apiserver | grep -v grep
```

Expected result:

'false' is equal to 'false'

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

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

Remediation: If using a Kubelet config file, edit the file to set authentication: anonymous: enabled to false. If using executable arguments, edit the kubelet service

file /etc/systemd/system/kubelet.service.d/10-kubeadm.conf on each worker node and set the below parameter in KUBELET_SYSTEM_PODS_ARGS variable.

```
--anonymous-auth=false
```

Based on your system, restart the kubelet service. For example:

```
systemctl daemon-reload
```

```
systemctl restart kubelet.service
```

Audit:

```
/bin/ps -fC kubelet
```

Audit Config:

```
/bin/cat /var/lib/kubelet/config.yaml
```

Expected result:

'false' is equal to 'false'

2) Ensure that the --authorization-mode argument is set to Webhook.

Audit

```
docker inspect kubelet | jq -e '.[0].Args[] | match("--authorization-mode=Webhook").string'
```

Returned Value: --authorization-mode=Webhook

Fix all of the following violations that were found against the ETCD:-

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

Do not use self-signed certificates for TLS. etcd is a highly-available key value store used by Kubernetes deployments for persistent storage of all of its REST API objects. These objects are sensitive in nature and should not be available to unauthenticated clients. You should enable the client authentication via valid certificates to secure the access to the etcd service.

Fix - Buildtime

Kubernetes

```
apiVersion: v1
```

```
kind: Pod
```

```
metadata:
```

```
annotations:  
  scheduler.alpha.kubernetes.io/critical-pod: ""  
creationTimestamp: null  
labels:  
  component: etcd  
  tier: control-plane  
name: etcd  
namespace: kube-system  
spec:  
  containers:  
    - command:  
      + - etcd  
      + - --auto-tls=true  
        image: k8s.gcr.io/etcd-amd64:3.2.18  
        imagePullPolicy: IfNotPresent  
        livenessProbe:  
          exec:  
            command:  
              - /bin/sh  
              - -ec  
              - ETCDCTL_API=3 etcdctl --endpoints=https://[192.168.22.9]:2379 --  
              cacert=/etc/kubernetes/pki/etcd/ca.crt  
              --cert=/etc/kubernetes/pki/etcd/healthcheck-client.crt --  
key=/etc/kubernetes/pki/etcd/healthcheck-client.key  
          get foo  
        failureThreshold: 8  
        initialDelaySeconds: 15  
        timeoutSeconds: 15  
      name: etcd-should-fail  
      resources: {}  
      volumeMounts:  
        - mountPath: /var/lib/etcd  
          name: etcd-data  
        - mountPath: /etc/kubernetes/pki/etcd  
          name: etcd-certs  
    hostNetwork: true  
    priorityClassName: system-cluster-critical  
    volumes:  
      - hostPath:  
          path: /var/lib/etcd  
          type: DirectoryOrCreate  
          name: etcd-data  
      - hostPath:  
          path: /etc/kubernetes/pki/etcd  
          type: DirectoryOrCreate  
          name: etcd-certs  
status: {}
```

Explanation:

```
candidate@cli:~$ kubectl delete sa/podrunner -n qa
serviceaccount "podrunner" deleted
candidate@cli:~$ kubectl config use-context KSCS00201
Switched to context "KSCS00201".
candidate@cli:~$ ssh kscs00201-master
Warning: Permanently added '10.240.86.194' (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@kscs00201-master:~# vim /etc/kubernetes/manifests/kube-apiserver.yaml
root@kscs00201-master:~# systemctl daemon-reload
root@kscs00201-master:~# systemctl restart kubelet.service
root@kscs00201-master:~# systemctl enable kubelet.service
root@kscs00201-master:~# systemctl status kubelet.service
● kubelet.service - kubelet: The Kubernetes Node Agent
   Loaded: loaded (/lib/systemd/system/kubelet.service; enabled; vendor preset: enabled)
     Drop-In: /etc/systemd/system/kubelet.service.d
               └─10-kubeadm.conf
   Active: active (running) since Fri 2022-05-20 14:19:31 UTC; 29s ago
     Docs: https://kubernetes.io/docs/home/
     Main PID: 134205 (kubelet)
       Tasks: 16 (limit: 76200)
      Memory: 39.5M
        CGroup: /system.slice/kubelet.service
                  └─134205 /usr/bin/kubelet --bootstrap-kubeconfig=/etc/kubernetes/bootstrap-kub
```

```
May 20 14:19:35 kscs00201-master kubelet[134205]: I0520 14:19:35.420825 134205 reconciler.>
May 20 14:19:35 kscs00201-master kubelet[134205]: I0520 14:19:35.420863 134205 reconciler.>
May 20 14:19:35 kscs00201-master kubelet[134205]: I0520 14:19:35.420907 134205 reconciler.>
May 20 14:19:35 kscs00201-master kubelet[134205]: I0520 14:19:35.420928 134205 reconciler.>
May 20 14:19:36 kscs00201-master kubelet[134205]: I0520 14:19:36.572353 134205 request.go:>
May 20 14:19:37 kscs00201-master kubelet[134205]: I0520 14:19:37.112347 134205 prober mana>
May 20 14:19:37 kscs00201-master kubelet[134205]: E0520 14:19:37.185076 134205 kubelet.go:>
May 20 14:19:37 kscs00201-master kubelet[134205]: I0520 14:19:37.645798 134205 kubelet.go:>
May 20 14:19:38 kscs00201-master kubelet[134205]: I0520 14:19:38.184062 134205 kubelet.go:>
May 20 14:19:40 kscs00201-master kubelet[134205]: I0520 14:19:40.036042 134205 prober_mana>
```

Lines 1-22/22 (END)

```
de Agent
et.service; enabled; vendor preset: enabled)
ce.d
```

5-20 14:19:31 UTC; 29s ago

```
trap-kubeconfig=/etc/kubernetes/bootstrap-kubelet.conf --kubeconfig=/etc/kubernetes/kubelet>
5]: I0520 14:19:35.420825 134205 reconciler.go:221] "operationExecutor.VerifyControllerAtt>
5]: I0520 14:19:35.420863 134205 reconciler.go:221] "operationExecutor.VerifyControllerAtt>
5]: I0520 14:19:35.420907 134205 reconciler.go:221] "operationExecutor.VerifyControllerAtt>
5]: I0520 14:19:35.420928 134205 reconciler.go:157] "Reconciler: start to sync state"
5]: I0520 14:19:36.572353 134205 request.go:665] Waited for 1.049946364s due to client-sid>
5]: I0520 14:19:37.112347 134205 prober_manager.go:255] "Failed to trigger a manual run" p>
5]: E0520 14:19:37.185076 134205 kubelet.go:1711] "Failed creating a mirror pod for" err=">
5]: I0520 14:19:37.645798 134205 kubelet.go:1693] "Trying to delete pod" pod="kube-system/>
5]: I0520 14:19:38.184062 134205 kubelet.go:1698] "Deleted mirror pod because it is outdat>
5]: I0520 14:19:40.036042 134205 prober_manager.go:255] "Failed to trigger a manual run" p>
~
~
lines 1-22/22 (END)
```

```
let.conf --kubeconfig=/etc/kubernetes/kubelet.conf --config=/var/lib/kubelet/config.yaml -->
o:221] "operationExecutor.VerifyControllerAttachedVolume started for volume \"kube-proxy\" >
o:221] "operationExecutor.VerifyControllerAttachedVolume started for volume \"lib-modules\" >
o:221] "operationExecutor.VerifyControllerAttachedVolume started for volume \"flannel-cfg\" >
o:157] "Reconciler: start to sync state"
65] Waited for 1.049946364s due to client-side throttling, not priority and fairness, request.go:255] "Failed to trigger a manual run" probe="Readiness"
711] "Failed creating a mirror pod for" err="pods \"kube-apiserver-kscs00201-master\" already exists"
693] "Trying to delete pod" pod="kube-system/kube-apiserver-kscs00201-master" podUID=bb91e1>
698] "Deleted mirror pod because it is outdated" pod="kube-system/kube-apiserver-kscs00201->
er.go:255] "Failed to trigger a manual run" probe="Readiness"
~
~
root@kscs00201-master:~# vim /var/lib/kubelet/config.yaml
apiVersion: kubelet.config.k8s.io/v1beta1
authentication:
  anonymous:
    enabled: false
  webhook:
    cacheTTL: 0s
    enabled: true
  x509:
    clientCAFile: /etc/kubernetes/pki/ca.crt
authorization:
  mode: Webhook
  webhook:
    cacheAuthorizedTTL: 0s
    cacheUnauthorizedTTL: 0s
cgroupDriver: systemd
clusterDNS:
~
~
root@kscs00201-master:~# vim /var/lib/kubelet/config.yaml
root@kscs00201-master:~# vim /var/lib/kubelet/config.yaml
root@kscs00201-master:~# vim /etc/kubernetes/manifests/etc.yaml
root@kscs00201-master:~# systemctl daemon-reload
root@kscs00201-master:~# systemctl restart kubelet.service
root@kscs00201-master:~# systemctl status kubelet.service
```

```
● kubelet.service - kubelet: The Kubernetes Node Agent
  Loaded: loaded (/lib/systemd/system/kubelet.service; enabled; vendor preset: enabled)
  Drop-In: /etc/systemd/system/kubelet.service.d
    └─10-kubeadm.conf
    Active: active (running) since Fri 2022-05-20 14:22:29 UTC; 4s ago
      Docs: https://kubernetes.io/docs/home/
    Main PID: 135849 (kubelet)
       Tasks: 17 (limit: 76200)
      Memory: 38.0M
        CGroup: /system.slice/kubelet.service
                  └─135849 /usr/bin/kubelet --bootstrap-kubeconfig=/etc/kubernetes/bootstrap-kub>
May 20 14:22:30 ksks00201-master kubelet[135849]: I0520 14:22:30.330232 135849 reconciler.>
May 20 14:22:30 ksks00201-master kubelet[135849]: I0520 14:22:30.330259 135849 reconciler.>
May 20 14:22:30 ksks00201-master kubelet[135849]: I0520 14:22:30.330304 135849 reconciler.>
May 20 14:22:30 ksks00201-master kubelet[135849]: I0520 14:22:30.330354 135849 reconciler.>
May 20 14:22:30 ksks00201-master kubelet[135849]: I0520 14:22:30.330378 135849 reconciler.>
May 20 14:22:30 ksks00201-master kubelet[135849]: I0520 14:22:30.330397 135849 reconciler.>
May 20 14:22:30 ksks00201-master kubelet[135849]: I0520 14:22:30.330415 135849 reconciler.>
May 20 14:22:30 ksks00201-master kubelet[135849]: I0520 14:22:30.330433 135849 reconciler.>
May 20 14:22:30 ksks00201-master kubelet[135849]: I0520 14:22:30.330452 135849 reconciler.>
May 20 14:22:30 ksks00201-master kubelet[135849]: I0520 14:22:30.330463 135849 reconciler.>
lines 1-22/22 (END)
May 20 14:22:30 ksks00201-master kubelet[135849]: I0520 14:22:30.330463 135849 reconciler.>
root@ksks00201-master:~#
root@ksks00201-master:~#
root@ksks00201-master:~#
root@ksks00201-master:~# exit
logout
Connection to 10.240.86.194 closed.
candidate@cli:~$
```

Question: 3

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.

**Answer: See the
Explanation below.**

Explanation:

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- <<EOF
apiVersion: v1
kind: Pod
metadata:
  name: pause
spec:
  containers:
    - name: pause
      image: k8s.gcr.io/pause
EOF
```

The output is similar to this:

```
Error from server (Forbidden): error when creating "STDIN": pods "pause" is forbidden: unable to
validate against any pod security policy: []
```

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

```
$ 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 ClusterRole named prevent-role, which uses the newly created Pod Security Policy prevent-privileged-policy.

```
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- <<EOF
```

```
apiVersion: v1
```

```
kind: Pod
```

```
metadata:
```

```
  name: pause
```

```
spec:
```

```
  containers:
```

```
    - name: pause
```

```
      image: k8s.gcr.io/pause
```

```
EOF
```

The output is similar to this:

```
Error from server (Forbidden): error when creating "STDIN": pods "pause" is forbidden: unable to validate against any pod security policy: []
```

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

```
apiVersion: rbac.authorization.k8s.io/v1
# This role binding allows "jane" to read pods in the "default" namespace.
# You need to already have a Role named "pod-reader" in that namespace.
kind: RoleBinding
metadata:
  name: read-pods
  namespace: default
subjects:
# You can specify more than one "subject"
- kind: User
  name: jane # "name" is case sensitive
apiGroup: rbac.authorization.k8s.io
```

```
roleRef:  
  # "roleRef" specifies the binding to a Role / ClusterRole  
  kind: Role #this must be Role or ClusterRole  
  name: pod-reader # this must match the name of the Role or ClusterRole you wish to bind to  
  apiGroup: rbac.authorization.k8s.io  
  
apiVersion: rbac.authorization.k8s.io/v1  
kind: Role  
metadata:  
  namespace: default  
  name: pod-reader  
rules:  
  - apiGroups: [""] # "" indicates the core API group  
    resources: ["pods"]  
    verbs: ["get", "watch", "list"]
```

Question: 4

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] $ | kubec  
t1 config use-context KS  
CH00201
```

Context

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

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.

Don't delete the existing
RoleBinding.



**Answer: See
explanation below.**

Explanation:

```
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:~$ 
```

Question: 5

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 12 days.
3. at maximum, a number of 8 old audit logs files are retained.
4. set the maximum size before getting rotated to 200MB

Edit and extend the basic policy to log:

1. namespaces changes at RequestResponse
2. Log the request body of secrets changes in the namespace kube-system.
3. Log all other resources in core and extensions at the Request level.
4. Log "pods/portforward", "services/proxy" at Metadata level.
5. Omit the Stage RequestReceived

All other requests at the Metadata level

Answer: See the explanation below:

Explanation:

Kubernetes auditing provides a security-relevant chronological set of records about a cluster. Kube-apiserver performs auditing. Each request on each stage of its execution generates an event, which is then pre-processed according to a certain policy and written to a backend. The policy determines what's recorded and the backends persist the records.

You might want to configure the audit log as part of compliance with the CIS (Center for Internet Security) Kubernetes Benchmark controls.

The audit log can be enabled by default using the following configuration in cluster.yml:

services:

 kube-api:

 audit_log:

 enabled: true

When the audit log is enabled, you should be able to see the default values at /etc/kubernetes/audit-policy.yaml

The log backend writes audit events to a file in [JSONlines](#) format. You can configure the log audit backend using the following kube-apiserver flags:

--audit-log-path specifies the log file path that log backend uses to write audit events. Not specifying this flag disables log backend. - means standard out

--audit-log-maxage defined the maximum number of days to retain old audit log files

--audit-log-maxbackup defines the maximum number of audit log files to retain

--audit-log-maxsize defines the maximum size in megabytes of the audit log file before it gets rotated

If your cluster's control plane runs the kube-apiserver as a Pod, remember to mount the hostPath to the location of the policy file and log file, so that audit records are persisted. For example:

 --audit-policy-file=/etc/kubernetes/audit-policy.yaml \

 --audit-log-path=/var/log/audit.log

Question: 6

Analyze and edit the given Dockerfile

FROM ubuntu:latest

RUN apt-get update -y

RUN apt-install nginx -y

COPY entrypoint.sh /

ENTRYPOINT ["/entrypoint.sh"]

USER ROOT

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

Analyze and edit the deployment manifest file

```
apiVersion: v1
kind: Pod
metadata:
  name: security-context-demo-2
spec:
  securityContext:
    runAsUser: 1000
  containers:
    - name: sec-ctx-demo-2
      image: gcr.io/google-samples/node-hello:1.0
      securityContext:
        runAsUser: 0
        privileged: True
        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

Answer: See the explanation below:

Explanation:

```
FROM debian:latest
```

```
MAINTAINER k@bogotobogo.com
```

```
# 1 - RUN
```

```
RUN apt-get update && 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 && 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: 7

Create a RuntimeClass named gvisor-rc using the prepared runtime handler named runsc.
Create a Pods of image Nginx in the Namespace server to run on the gVisor runtime class

Answer: See the explanation below:

Explanation:

Install the Runtime Class for gVisor

```
{ # Step 1: Install a RuntimeClass
cat <<EOF | kubectl apply -f -
apiVersion: node.k8s.io/v1beta1
kind: RuntimeClass
metadata:
  name: gvisor
handler: runsc
EOF
}
```

Create a Pod with the gVisor Runtime Class

```
{ # Step 2: Create a pod
cat <<EOF | kubectl apply -f -
apiVersion: v1
kind: Pod
metadata:
  name: nginx-gvisor
spec:
  runtimeClassName: gvisor
```

```
containers:  
- name: nginx  
  image: nginx  
EOF  
}
```

Verify that the Pod is running

```
{ # Step 3: Get the pod  
kubectl get pod nginx-gvisor -o wide  
}
```

Question: 8

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

Cluster	Master node	Worker node
KSSH00301	kssh00301 -master	kssh00301 -worker1

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

```
[candidate@cli] $ | kubectl config use-context KSSH00301
```

Task

Create a NetworkPolicy named pod-access to restrict access to Pod users-service running in namespace dev-team.

Only allow the following Pods to connect to Pod users-service:

- Pods in the namespace qa
- Pods with label environment: testing, in any namespace

Make sure to apply the NetworkPolicy.



You can find a skeleton manifest file at
`/home/candidate/KSSH00301/network-policy.yaml`



Answer: See explanation below.

Explanation:

```
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: []
```

```
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:~$ █
```

Question: 9

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.

**Answer: See
explanation below.**

Explanation:

```
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-dev-0v210johnn903-master0]
ok: [kubernetes-dev-0v210johnn903-master1]
ok: [kubernetes-dev-0v210johnn903-master2]

[TASK [kubernetes/master : Create kubeadm config] ****
Tuesday 03 September 2019  07:14:21 +0000 (0:00:00.560)      0:21:59.046 ****
changed: [kubernetes-dev-0v210johnn903-master0]
changed: [kubernetes-dev-0v210johnn903-master1]
changed: [kubernetes-dev-0v210johnn903-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-dev-0v210johnn903-master0]: FAILED! => {"attempts": 3, "changed": true, "cmd": ["timeout", "-k", "600s", "600s", "/opt/bin/kubeadm", "init", "--config=/etc/kubernetes/kube-adm-config.yaml", "--ignore-preflight-errors=all", "--skip-phases=addon/coredns", "--experimental-upload-certs", "--cert-file-key=ecab4f2d9e5b2eddb072c1a977d5c8409b5d4a09842rcba3fe4e5cc0b", "delta": "+0:02:02.44906", "end": "2019-09-03 07:23:13.97186", "failed_when_result": true, "msg": "non-zero return code", "rc": 1, "start": "2019-09-03 07:21:52.231", "stdout": "[WARNING Port-443]: Port 443 is in use\\n[WARNING Port-10251]: Port 10251 is in use\\n[WARNING FileAvailable--etc-kubernetes-manifests-kube-apiserver.yaml]: /etc/kubernetes/manifests/kube-apiserver.yaml already exists\\n[WARNING FileAvailable--etc-kubernetes-manifests-kube-controller-manager.yaml]: /etc/kubernetes/manifests/kube-controller-manager.yaml already exists\\n[WARNING FileAvailable--etc-kubernetes-manifests-kube-scheduler.yaml]: /etc/kubernetes/manifests/kube-scheduler.yaml already exists\\n[WARNING IsSocketSystemCheck]: detected \"cgroups\" as the Docker cgroup driver. The recommended driver is \"systemd\". Please follow the guide at https://kubernetes.io/docs/setup/ci/intro[WARNING Port-10250]: port 10250 is in use\\n[WARNING ExecutedPreflight]: detected \"cgroups\" as the Docker cgroup driver. The recommended driver is \"systemd\". Please follow the guide at https://kubernetes.io/docs/setup/ci/\", "stderr_lines": ["[WARNING Port-443]: Port 443 is in use", "[WARNING Port-10251]: Port 10251 is in use", "[WARNING FileAvailable--etc-kubernetes-manifests-kube-apiserver.yaml]: /etc/kubernetes/manifests/kube-apiserver.yaml already exists", "[WARNING FileAvailable--etc-kubernetes-manifests-kube-controller-manager.yaml]: /etc/kubernetes/manifests/kube-controller-manager.yaml already exists", "[WARNING FileAvailable--etc-kubernetes-manifests-kube-scheduler.yaml]: /etc/kubernetes/manifests/kube-scheduler.yaml already exists", "[WARNING IsSocketSystemCheck]: detected \"cgroups\" as the Docker cgroup driver. The recommended driver is \"systemd\". Please follow the guide at https://kubernetes.io/docs/setup/ci/"], "stdout": "[init] Using Kubernetes version: v1.16.4+6nfipgj 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 configuration to file \"/var/lib/kubelet/kubelet_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 certificate authority\\n[certs] Using existing ca certificate authority\\n[certs] Using existing apiserver certificate and key on disk\\n[certs]
```

Question: 10

On the Cluster worker node, enforce the prepared AppArmor profile
#include <tunables/global>

```
profile nginx-deny flags=(attach_disconnected) {  
    #include <abstractions/base>
```

file,

```
# Deny all file writes.  
deny /** w,  
}  
EOF'
```

Edit the prepared manifest file to include the AppArmor profile.

```
apiVersion: v1
kind: Pod
metadata:
  name: apparmor-pod
spec:
  containers:
    - name: apparmor-pod
      image: nginx
```

Finally, apply the manifests files and create the Pod specified on it.

Verify: Try to make a file inside the directory which is restricted.

**Answer: See
explanation below.**

Explanation:

```
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
```

Question: 11

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

Answer: See the explanation below:

Explanation:

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: 12

a. 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

b. 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

**Answer: Answer: See
the explanation
below:**

Explanation:

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

Navigate to your:

Project's Operations > Kubernetes page, for a project-level cluster.

Group's Kubernetes page, for a group-level cluster.

Admin Area > Kubernetes page, for an instance-level cluster.

Click Add Kubernetes cluster.

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

Kubernetes cluster name (required) - The name you wish to give the cluster.

Environment scope (required) - The [associated environment](#) to this cluster.

API URL (required) - It's the URL that GitLab uses to access the Kubernetes API. Kubernetes exposes several APIs, we want the "base" URL that is common to all of them. For example, <https://kubernetes.example.com> rather than <https://kubernetes.example.com/api/v1>.

Get the API URL by running this command:

```
kubectl cluster-info | grep -E 'Kubernetes master|Kubernetes control plane' | awk '/http/{print $NF}'  
CA certificate (required) - A valid Kubernetes certificate is needed to authenticate to the cluster. We  
use the certificate created by default.
```

List the secrets with kubectl get secrets, and one should be named similar to default-token-xxxxx.

Copy that token name for use below.

Get the certificate by running this command:

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

Question: 13

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

Answer: Send us your suggestion on it.

Question: 14

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

Cluster	Master node	Worker node
KSRS001 01	ksrs00101 -master	ksrs00101- worker1

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

```
[candidate@cli] $ | kubec  
t1 config use-context KS  
RS00101
```

You may use your browser to open **one additional tab** to access Falco's documentation.

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

- sysdig
- falco

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,processName
```

The following example shows a properly formatted incident file:

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

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



Make sure to store the incident file on the cluster's worker node.



Answer: See explanation below.

Explanation:

```
candidate@cli:~$ kubectl config use-context KSRS00101
Switched to context "KSRS00101".
candidate@cli:~$ ssh ksrs00101-worker1
Warning: Permanently added '10.240.86.96' (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@ksrs00101-worker1:~# falco
falco          falco-driver-loader
root@ksrs00101-worker1:~# ls -l /etc/falco/
total 200
-rw-r--r-- 1 root root 12399 Jan 31 16:06 aws_clouptrail_rules.yaml
-rw-r--r-- 1 root root 11384 Jan 31 16:06 falco.yaml
-rw-r--r-- 1 root root 1136 Jan 31 16:06 falco_rules.local.yaml
-rw-r--r-- 1 root root 132112 Jan 31 16:06 falco_rules.yaml
-rw-r--r-- 1 root root 27289 Jan 31 16:06 k8s_audit_rules.yaml
drwxr-xr-x 2 root root 4096 Feb 16 01:07 rules.available
drwxr-xr-x 2 root root 4096 Jan 31 16:28 rules.d
root@ksrs00101-worker1:~# vim /etc/falco/falco_rules.local.yaml
```

```
# Copyright (C) 2019 The Falco Authors.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#
#####
# Your custom rules!
#####

# Add new rules, like this one
# - rule: The program "sudo" is run in a container
#   desc: An event will trigger every time you run sudo in a container
#   condition: evt.type = execve and evt.dir=< and container.id != host and proc.name = sudo
#   output: "Sudo run in container (user=%user.name %container.info parent=%proc.pname cmdline=%proc.cmdline)"
#   priority: ERROR
#   tags: [users, container]

# Or override/append to any rule, macro, or list from the Default Rules
- rule: Container Drift Detected (chmod)
  desc: New executable created in a container due to chmod
  condition: >
    evt.type in (open,openat,create) 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
  priority: ERROR

root@ksrs00101-worker1:~# vim /etc/falco/falco_rules.local.yaml
root@ksrs00101-worker1:~# systemctl status falco.service
● falco.service - Falco Runtime Security
    Loaded: loaded (/lib/systemd/system/falco.service; disabled; vendor preset: enabled)
      Active: inactive (dead)
root@ksrs00101-worker1:~# systemctl enable falco.service
Created symlink /etc/systemd/system/multi-user.target.wants/falco.service → /lib/systemd/system/falco.service.
root@ksrs00101-worker1:~# systemctl start falco.service
root@ksrs00101-worker1:~# exit
logout
Connection to 10.240.86.96 closed.
candidate@cli:~$ ssh ksrs00101-worker1
Last login: Fri May 20 15:59:48 2022 from 10.240.86.88
root@ksrs00101-worker1:~# vim /etc/falco/falco.yaml
```

```
# When using json output, whether or not to include the "tags" property
# itself in the json output. If set to true, outputs caused by rules
# with no tags will have a "tags" field set to an empty array. If set to
# false, the "tags" field will not be included in the json output at all.
json_include_tags_property: true

# Send information logs to stderr and/or syslog Note these are *not* security
# notification logs! These are just Falco lifecycle (and possibly error) logs.
log_stderr: true
log_syslog: true
log_file: /opt/KSRS00101/alerts/details

# Minimum log level to include in logs. Note: these levels are
# separate from the priority field of rules. This refers only to the
# log level of falco's internal logging. Can be one of "emergency",
# "alert", "critical", "error", "warning", "notice", "info", "debug".
log_level: info

root@ksrs00101-worker1:~# vim /etc/falco/falco.yaml
root@ksrs00101-worker1:~# grep log /etc/falco/falco.yaml
# cloudtrail log files.
# If true, the times displayed in log messages and output messages
# Send information logs to stderr and/or syslog Note these are *not* security
# notification logs! These are just Falco lifecycle (and possibly error) logs.
log_stderr: true
log_syslog: true
log_file: /opt/KSRS00101/alerts/details
# Minimum log level to include in logs. Note: these levels are
# log level of falco's internal logging. Can be one of "emergency",
log_level: info
# - log: log a DEBUG message noting that the buffer was full
# Notice it is not possible to ignore and log/alert messages at the same time.
# The rate at which log/alert messages are emitted is governed by a
# - log
# The timeout error will be reported to the log according to the above log_* settings.
syslog_output:
# - logging (alternate method than syslog):
#     program: logger -t falco-test
# this information will be logged, however the main Falco daemon will not be stopped.
root@ksrs00101-worker1:~# systemctl restart falco.service
root@ksrs00101-worker1:~# exit
logout
Connection to 10.240.86.96 closed.
candidate@cli:~$ █
```

Question: 15

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.

**Answer: See the
Explanation below.**

Explanation:

se 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-rosource-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: 16

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:-

a. Ensure that the RotateKubeletServerCertificate argument is set to true.

b. Ensure that the admission control plugin PodSecurityPolicy is set.

c. Ensure that the --kubelet-certificate-authority argument is set as appropriate.

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

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

b. Ensure that the --authorization-mode argument is set to Webhook.

Fix all of the following violations that were found against the ETCD:-

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

Hint: Take the use of Tool Kube-Bench

Answer: See the Explanation below.

Explanation:

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=<ca-string>
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 master
node 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 master
node and either remove the --peer-auto-tls parameter or set it to false.
--peer-auto-tls=false

Question: 17

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 manifest, it should get failed.

POD Manifest:

```
apiVersion: v1
kind: Pod
metadata:
  name:
spec:
  containers:
  - name:
    image:
    volumeMounts:
    - name:
      mountPath:
  volumes:
  - name:
    secret:
      secretName:
```

**Answer: See the
Explanation below:**

Explanation:

```
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: 18

Given an existing Pod named nginx-pod running in the namespace test-system, fetch the service-account-name used and put the content in /candidate/KSC00124.txt

Create a new Role named dev-test-role in the namespace test-system, which can perform update operations, on resources of type namespaces.

Create a new RoleBinding named dev-test-role-binding, which binds the newly created Role to the Pod's ServiceAccount (found in the Nginx pod running in namespace test-system).

**Answer: See
explanation below.**

Explanation:

```
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-8fdb-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
  *          []                  [*]
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:~$ 
```

Question: 19

Enable audit logs in the cluster, To Do so, enable the log backend, and ensure that

1. logs are stored at /var/log/kubernetes/kubernetes-logs.txt.
2. Log files are retained for 5 days.
3. at maximum, a number of 10 old audit logs files are retained.

Edit and extend the basic policy to log:

1. Cronjobs changes at RequestResponse
2. Log the request body of deployments changes in the namespace kube-system.
3. Log all other resources in core and extensions at the Request level.
4. Don't log watch requests by the "system:kube-proxy" on endpoints or

**Answer: See
explanation below.**

Explanation:

```
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:~$ █
```

Question: 20

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

Create a Pods of image alpine:3.13.2 in the Namespace default to run on the gVisor runtime class.

Answer: See the explanation below:

Explanation:

```
[ 0.000000] Starting gVisor...
[ 0.183366] Creating cloned children...
[ 0.290397] Moving files to filing cabinet...
[ 0.392925] Letting the watchdogs out...
[ 0.452958] Digging up root...
[ 0.937597] Gathering forks...
[ 1.095681] Daemonizing children...
[ 1.306448] Rewriting operating system in Javascript...
[ 1.514936] Reading process obituaries...
[ 1.589958] Waiting for children...
[ 1.892298] Segmenting fault lines...
[ 1.974948] Ready!
```

Question: 21

Create a network policy named allow-np, that allows pod in the namespace staging to connect to port 80 of other pods in the same namespace.

Ensure that Network Policy:-

1. Does not allow access to pod not listening on port 80.
2. Does not allow access from Pods, not in namespace staging.

Answer: See the explanation below:

Explanation:

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: network-policy
spec:
  podSelector: {} #selects all the pods in the namespace deployed
  policyTypes:
  - Ingress
  ingress:
  - ports:      #in input traffic allowed only through 80 port only
    - protocol: TCP
    port: 80
```

Question: 22

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.

Answer: See explanation below.

Explanation:

```
$ kubectl get ing -n <namespace-of-ingress-resource>
NAME      HOSTS      ADDRESS      PORTS      AGE
cafe-ingress  cafe.com  10.0.2.15  80          25s
```

```
$ kubectl describe ing <ingress-resource-name> -n <namespace-of-ingress-resource>
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 (<none>)
  /coffee  coffee-svc:80 (<none>)
```

Annotations:

```
kubectl.kubernetes.io/last-applied-configuration:
{"apiVersion":"networking.k8s.io/v1","kind":"Ingress","metadata":{"annotations":{},"name":"cafe-ingress","namespace":"default","selfLink":"/apis/networking/v1/namespaces/default/ingresses/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":"169.48.142.11"}]}}}
```

0"}]}{}

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 <namespace-of-ingress-controller>				
NAME READY STATUS RESTARTS AGE				
ingress-nginx-controller-67956bf89d-fv58j 1/1 Running 0 1m				

\$ kubectl logs -n <namespace> ingress-nginx-controller-67956bf89d-fv58j

NGINX Ingress controller

Release: 0.14.0
Build: git-734361d
Repository: <https://github.com/kubernetes/ingress-nginx>

....

Question: 23

Secrets stored in the etcd is not secure at rest, you can use the etcdctl command utility to find the secret value

for e.g:-

ETCDCTL_API=3 etcdctl get /registry/secrets/default/cks-secret --cacert="ca.crt" --cert="server.crt" --key="server.key"

Output

```
/registry/secrets/default/cks-secret
k8s

key1: secret
key2: topsecret

cks-secret@default":*567fcb53f-6b58-4fee-9f12-5737c764be742***:*
kubectl-create[update]*****[fieldsV1:9
  "key1": "supersecret", "key2": "topsecret", "f:type": {}]
key1: supersecret
key2: topsecret[Opaque]*

Visible
```

Using the Encryption Configuration, Create the manifest, which secures the resource secrets using the provider AES-CBC and identity, to encrypt the secret-data at rest and ensure all secrets are encrypted with the new configuration.

Answer: See explanation below.

Explanation:

ETCD secret encryption can be verified with the help of etcdctl command line utility.

ETCD secrets are stored at the path /registry/secrets/\$namespace/\$secret on the master node.

The below command can be used to verify if the particular ETCD secret is encrypted or not.

```
# ETCDCTL_API=3 etcdctl get /registry/secrets/default/secret1 [...] | hexdump -C
```

Question: 24

Service is running on port 389 inside the system, find the process-id of the process, and stores the names of all the open-files inside the /candidate/KH77539/files.txt, and also delete the binary.

Answer: See explanation below.

Explanation:

```
root# netstat -ltnup
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address          Foreign Address          State   PID/Program name
tcp    0    0 127.0.0.1:17600      0.0.0.0:*      LISTEN  1293/dropbox
tcp    0    0 127.0.0.1:17603      0.0.0.0:*      LISTEN  1293/dropbox
tcp    0    0 0.0.0.0:22        0.0.0.0:*      LISTEN  575/sshd
tcp    0    0 127.0.0.1:9393      0.0.0.0:*      LISTEN  900/perl
tcp    0    0 :::80            ::*:          LISTEN  9583/docker-proxy
tcp    0    0 :::443           ::*:          LISTEN  9571/docker-proxy
udp    0    0 0.0.0.0:68        0.0.0.0:*      8822/dhcpcd
...
root# netstat -ltnup | grep ':22'
tcp    0    0 0.0.0.0:22        0.0.0.0:*      LISTEN  575/sshd
```

The [ss](#) command is the replacement of the netstat command.

Now let's see how to use the ss command to see which process is listening on port 22:

```
root# ss -ltnup 'sport = :22'
```

```
Netid State Recv-Q Send-Q Local Address:Port Peer Address:Port
```

```
tcp  LISTEN 0    128   0.0.0.0:22      0.0.0.0:*      users:(:"sshd",pid=575,fd=3))
```

Question: 25

Use the kubesec docker images to scan the given YAML manifest, edit and apply the advised changes, and passed with a score of 4 points.

```
kubesec-test.yaml
apiVersion: v1
kind: Pod
metadata:
  name: kubesec-demo
spec:
  containers:
  - name: kubesec-demo
    image: gcr.io/google-samples/node-hello:1.0
```

```
securityContext:  
  readOnlyRootFilesystem: true
```

Hint: docker run -i kubesec/kubesec:512c5e0 scan /dev/stdin < kubesec-test.yaml

**Answer: See
explanation below.**

Explanation:

```
kubesec scan k8s-deployment.yaml  
cat <<EOF > kubesec-test.yaml  
apiVersion: v1  
kind: Pod  
metadata:  
  name: kubesec-demo  
spec:  
  containers:  
    - name: kubesec-demo  
      image: gcr.io/google-samples/node-hello:1.0  
      securityContext:  
        readOnlyRootFilesystem: true  
EOF  
kubesec scan kubesec-test.yaml
```

docker run -i kubesec/kubesec:512c5e0 scan /dev/stdin < kubesec-test.yaml

```
kubesec http 8080 &  
[1] 12345  
{"severity":"info","timestamp":"2019-05-  
12T11:58:34.662+0100","caller":"server/server.go:69","message":"Starting HTTP server on port  
8080"}
```

```
curl -sSX POST --data-binary @test/asset/score-0-cap-sys-admin.yml http://localhost:8080/scan  
[  
 {  
   "object": "Pod/security-context-demo.default",  
   "valid": true,  
   "message": "Failed with a score of -30 points",  
   "score": -30,  
   "scoring": {  
     "critical": [  
       {  
         "selector": "containers[] .securityContext .capabilities .add == SYS_ADMIN",  
         "reason": "CAP_SYS_ADMIN is the most privileged capability and should always be avoided"  
       },  
       {  
         "selector": "containers[] .securityContext .runAsNonRoot == true",  
         "reason": "Force the running image to run as a non-root user to ensure least privilege"
```

```
},  
// ...
```

Question: 26

Using the runtime detection tool Falco, Analyse the container behavior for at least 20 seconds, using filters that detect newly spawning and executing processes in a single container of Nginx. store the incident file art /opt/falco-incident.txt, containing the detected incidents. one per line, in the format [timestamp],[uid],[processName]

Answer: Send us your feedback on it.

Question: 27

Cluster: qa-cluster

Master node: master Worker node: worker1

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

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

Task:

Create a NetworkPolicy named restricted-policy to restrict access to Pod product running in namespace dev.

Only allow the following Pods to connect to Pod products-service:

1. Pods in the namespace qa
2. Pods with label environment: stage, in any namespace

Answer: See the Explanation below.

Explanation:

```
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: []
```

```
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:~$ █
```

Reference: <https://kubernetes.io/docs/concepts/services-networking/network-policies/>

Question: 28

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



Cluster	Master node	Worker node
KSSC00202	kssc00202-master	kssc00202-worker1

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

```
[candidate@cli] $ kubectl config use-context KSSC00202
```

Context

A container image scanner is set up on the cluster, but it's not yet fully integrated into the cluster's configuration. When complete, the container image scanner shall scan for and reject the use of vulnerable images.

Task

You have to complete the entire task on the cluster's master node, where all services and files have been prepared and placed.



Given an incomplete configuration in directory /etc/kubernetes/epconfig and a functional container image scanner with HTTPS endpoint https://wakanda.local:8081 /image_policy :

1. Enable the necessary plugins to create an image policy
2. Validate the control configuration and change it to an implicit deny
3. Edit the configuration to point to the provided HTTPS endpoint correctly

Finally, test if the configuration is working by trying to deploy the vulnerable resource /root/KSSC00202/vulnerable-resource.yml.

You can find the container image scanner's log file at
`/var/log/imagepolicy/acme.log`



Answer: See the explanation below

Explanation:

```
Switched to context "KSSC00202".
candidate@cli:~$ ssh kssc00202-master
Warning: Permanently added '10.177.80.12' (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@kssc00202-master:~# ls /etc/kubernetes/epconfig/
admission_configuration.json  apiserver-client-key.pem  apiserver-client.pem  kubeconfig.yaml  webhook-key.pem  webhook.pem
root@kssc00202-master:~# vim /etc/kubernetes/epconfig/admission_configuration.json
```

```
"imagePolicy": {  
    "kubeConfigFile": "/etc/kubernetes/epconfig/kubeconfig.yaml",  
    "allowTTL": 50,  
    "denyTTL": 50,  
    "retryBackoff": 500,  
    "defaultAllow": false}
```

```
root@kssc00202-master:~# vim /etc/kubernetes/epconfig/admission_configuration.json  
root@kssc00202-master:~# vim /etc/kubernetes/epconfig/admission_configuration.json  
root@kssc00202-master:~# vim /etc/kubernetes/epconfig/kubeconfig.yaml
```

```
apiVersion: v1  
clusters:  
- cluster:  
    certificate-authority: /etc/kubernetes/epconfig/webhook.pem # CA for verifying the remote service.  
    server: https://wakanda.local:8081/image_policy  
    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: /etc/kubernetes/epconfig/apiserver-client.pem  
    client-key: /etc/kubernetes/epconfig/apiserver-client-key.pem
```

```
root@kssc00202-master:~# vim /etc/kubernetes/epconfig/admission_configuration.json  
root@kssc00202-master:~# vim /etc/kubernetes/epconfig/admission_configuration.json  
root@kssc00202-master:~# vim /etc/kubernetes/epconfig/kubeconfig.yaml  
root@kssc00202-master:~# vim /etc/kubernetes/manifests/kube-apiserver.yaml p
```

```
apiVersion: v1
kind: Pod
metadata:
  annotations:
    kubeadm.kubernetes.io/kube-apiserver.advertise-address.endpoint: 10.177.80.12:6443
  creationTimestamp: null
  labels:
    component: kube-apiserver
    tier: control-plane
  name: kube-apiserver
  namespace: kube-system
spec:
  containers:
    - command:
        - kube-apiserver
        - --advertise-address=10.177.80.12
        - --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-
      "/etc/kubernetes/manifests/kube-apiserver.yaml" 135L, 4626C
```

```
root@kssc00202-master:~# vim /etc/kubernetes/manifests/kube-apiserver.yaml p
2 files to edit
root@kssc00202-master:~# rm -f p
root@kssc00202-master:~# vim /etc/kubernetes/manifests/kube-apiserver.yaml █
```

```
apiVersion: v1
kind: Pod
metadata:
  annotations:
    kubeadm.kubernetes.io/kube-apiserver.advertise-address.endpoint: 10.177.80.12:6443
  creationTimestamp: null
  labels:
    component: kube-apiserver
    tier: control-plane
  name: kube-apiserver
  namespace: kube-system
spec:
  containers:
    - command:
        - kube-apiserver
        - --advertise-address=10.177.80.12
        - --allow-privileged=true
        - --authorization-mode=Node,RBAC
        - --client-ca-file=/etc/kubernetes/pki/ca.crt
        - --enable-admission-plugins=NodeRestriction,ImagePolicyWebHook
        - --admission-control-config-file=/etc/kubernetes/epconfig/admin.conf
        - --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
```

```
root@kssc00202-master:~# rm -f p
root@kssc00202-master:~# vim /etc/kubernetes/manifests/kube-apiserver.yaml
root@kssc00202-master:~# systemctl daemon-reload
root@kssc00202-master:~#
root@kssc00202-master:~#
root@kssc00202-master:~#
root@kssc00202-master:~#
root@kssc00202-master:~#
root@kssc00202-master:~#
root@kssc00202-master:~# systemctl restart kubelet.service
root@kssc00202-master:~# systemctl enable kubelet.service
root@kssc00202-master:~#
root@kssc00202-master:~#
root@kssc00202-master:~#
root@kssc00202-master:~#
root@kssc00202-master:~# ls
KSSC00202 snap
root@kssc00202-master:~# cat KSSC00202/vulnerable-resource.yml
```

```
KSSC00202 snap
root@kssc00202-master:~# cat KSSC00202/vulnerable-resource.yml
---
apiVersion: v1
kind: ReplicationController
metadata:
  name: nginx-latest
spec:
  replicas: 1
  selector:
    app: nginx-latest
  template:
    metadata:
      name: nginx-latest
      labels:
        app: nginx-latest
    spec:
      containers:
        - name: nginx-latest
          image: nginx
          ports:
            - containerPort: 80
root@kssc00202-master:~# kubectl create -f KSSC00202/vulnerable-resource.yml
```

```
root@kssc00202-master:~# kubectl create -f KSSC00202/vulnerable-resource.yml
The connection to the server 10.177.80.12:6443 was refused - did you specify the right host or port?
root@kssc00202-master:~# kubectl get pods
The connection to the server 10.177.80.12:6443 was refused - did you specify the right host or port?
root@kssc00202-master:~# ls -al .kube/
total 20
drwxr-xr-x  3 root root 4096 Aug  3 04:07 .
drwx-----  9 root root 4096 Oct 11 15:36 ..
drwxr-x---  4 root root 4096 Aug  3 04:07 cache
-rw-r--r--  1 root root 5636 Aug  3 04:07 config
root@kssc00202-master:~# crictl ps -a
```

```
012ea8587130e      a634548d10b03      2 months ago      Exited      kube-proxy      0      1460a9f
a0f1e0      kube-proxy-cmjb5
405227dfa49d0      aebe758cef4cd      2 months ago      Exited      etcd      0      cfb6522
e720fb      etcd-kssc00202-master
root@kssc00202-master:~# ls -al .kube/ | grep kube-api
root@kssc00202-master:~# crictl ps -a | grep kube-api
WARN[0000] runtime connect using default endpoints: [unix:///var/run/dockershim.sock unix:///run/containerd/containerd.sock unix:///run/crio/crio.sock unix:///var/run/cri-dockerd.sock]. As the default settings are now deprecated, you should set the endpoint instead.
ERROR[0000] unable to determine runtime API version: rpc error: code = Unavailable desc = connection error: desc = "transport: Error while dialing dial unix /var/run/dockershim.sock: connect: no such file or directory"
WARN[0000] image connect using default endpoints: [unix:///var/run/dockershim.sock unix:///run/containerd/containerd.sock unix:///run/crio/crio.sock unix:///var/run/cri-dockerd.sock]. As the default settings are now deprecated, you should set the endpoint instead.
ERROR[0000] unable to determine image API version: rpc error: code = Unavailable desc = connection error: desc = "transport: Error while dialing dial unix /var/run/dockershim.sock: connect: no such file or directory"
a003b3fdb61c      d3377ffb7177c      30 seconds ago      Exited      kube-apiserver      3      2dad64e
984a91      kube-apiserver-kssc00202-master
5e70b9a70f9ed      d3377ffb7177c      7 hours ago      Exited      kube-apiserver      0      68a9f31
6c2559      kube-apiserver-kssc00202-master
root@kssc00202-master:~#
root@kssc00202-master:~#
root@kssc00202-master:~#
root@kssc00202-master:~# exit
logout
Connection to 10.177.80.12 closed.
candidate@cli:~$
```

Question: 29

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 namespace prod and 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 be privileged in any way must be treated as potentially not stateless or not immutable.

Answer: See the explanation below

Explanation:

```
candidate@cli:~$ kubectl config use-context KSR00501
Switched to context "KSR00501".
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://11143682c400984c9faf3dffle056d4b00a7eb1de007fe1834be0a84fa146e18
  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|ReadOnlyFile
system'
  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
```

Reference: <https://kubernetes.io/docs/concepts/policy/pod-security-policy/>
<https://cloud.google.com/architecture/best-practices-for-operating-containers>

Question: 30

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

Answer: See the explanation below

Explanation:

Create psp to disallow privileged container

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: deny-access-role
rules:
- apiGroups: ['policy']
  resources: ['podsecuritypolicies']
  verbs:  ['use']
  resourceNames:
  - "deny-policy"
k create sa psp-denial-sa -n development
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:
- kind: ServiceAccount
  name: psp-denial-sa
    namespace: development
```

Explanation

```
master1 $ 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 development

```
master1 $ 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
```

master1 \$ k apply -f psp.yaml
master1 \$ k apply -f cr1.yaml
master1 \$ k apply -f cb1.yaml

Reference: <https://kubernetes.io/docs/concepts/policy/pod-security-policy/>

Question: 31

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

Answer: See the explanation below

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: {}

```

Reference: <https://kubernetes.io/docs/concepts/policy/pod-security-policy/>

Question: 32

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.

Answer: See the explanation below

Explanation:

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

- level: RequestResponse
 - userGroups: ["system:nodes"]
- level: Request
 - resources:
 - group: "" # core API group
 - resources: ["persistentvolumes"]
 - namespaces: ["frontend"]
 - level: Metadata
 - resources:
 - group: ""
 - resources: ["configmaps", "secrets"]

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

Add these

- --audit-policy-file=/etc/kubernetes/log-policy/audit-policy.yaml
- --audit-log-path=/var/log/kubernetes/logs.txt
- --audit-log-maxage=5
- --audit-log-maxbackup=10

Explanation

```
[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 & 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
```

Note: log volume & policy volume is already mounted in vim /etc/kubernetes/manifests/kube-apiserver.yaml so no need to mount it.

Reference: <https://kubernetes.io/docs/tasks/debug-application-cluster/audit/>

Question: 33

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] $ | kubec  
t1 config use-context KS  
SH00401
```

Context

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

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.

Answer: See the explanation below

Explanation:

```
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
```

Reference: <https://kubernetes.io/docs/tutorials/clusters/apparmor/>

Question: 34

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

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

Context:

A pod fails to run because of an incorrectly specified ServiceAccount

Task:

Create a new service account named backend-qa in an existing namespace qa, which must not have access to any secret.

Edit the frontend pod yaml to use backend-qa service account

Note: You can find the frontend pod yaml at /home/cert_masters/frontend-pod.yaml

Answer: See the explanation below

Explanation:

```
[desk@cli] $ k create sa backend-qa -n qa
sa/backend-qa created
```

```
[desk@cli] $ k get role,rolebinding -n qa
No resources found in qa namespace.
```

```
[desk@cli] $ k create role backend -n qa --resource pods, namespaces, configmaps --verb list
# No access to secret
```

```
[desk@cli] $ k create rolebinding backend -n qa --role backend --serviceaccount qa:backend-qa
```

```
[desk@cli] $ vim /home/cert_masters/frontend-pod.yaml
apiVersion: v1
kind: Pod
```

```
metadata:  
  name: frontend  
spec:  
  serviceAccountName: backend-qa # Add this  
  image: nginx  
  name: frontend  
[desk@cli] $ k apply -f /home/cert_masters/frontend-pod.yaml  
pod created  
[desk@cli] $ k create sa backend-qa -n qa  
serviceaccount/backend-qa created
```

```
[desk@cli] $ k get role,rolebinding -n qa  
No resources found in qa namespace.
```

```
[desk@cli] $ k create role backend -n qa --resource pods, namespaces, configmaps --verb list  
role.rbac.authorization.k8s.io/backend created
```

```
[desk@cli] $ k create rolebinding backend -n qa --role backend --serviceaccount qa:backend-qa  
rolebinding.rbac.authorization.k8s.io/backend created
```

```
[desk@cli] $ vim /home/cert_masters/frontend-pod.yaml  
apiVersion: v1  
kind: Pod  
metadata:  
  name: frontend  
spec:  
  serviceAccountName: backend-qa # Add this  
  image: nginx  
  name: frontend  
[desk@cli] $ k apply -f /home/cert_masters/frontend-pod.yaml  
pod/frontend created
```

<https://kubernetes.io/docs/tasks/configure-pod-container/configure-service-account/>

Question: 35

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.

Answer: See the explanation below

Explanation:

```
$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
$kill -1 <PID of falco>
```

Explanation

```
[desk@cli] $ ssh node01
[node01@cli] $ vim /etc/falco/falco_rules.yaml
search for Container Drift Detected & 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
```

```
file_output:  
  enabled: true  
  keep_alive: false  
  filename: /home/cert_masters/report
```

send HUP signal to falco process to re-read the configuration

```
root@node01:/etc/falco# ps -ef | grep falco  
root      10127     1  1 17:13 ?        00:00:35 /usr/bin/falco --pidfile=/var/run/falco.pid -c /etc/falco/falco.yaml  
root      30938 20175  0 17:55 pts/1    00:00:00 grep falco  
root@node01:/etc/falco# kill -1 10127
```

Reference:

<https://falco.org/docs/alerts/>

<https://falco.org/docs/rules/supported-fields/>

Question: 36

Cluster: dev

Master node: master1

Worker node: worker1

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

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

Task:

Retrieve the content of the existing secret named adam in the safe namespace.

Store the username field in a file names /home/cert-masters/username.txt, and the password field in a file named /home/cert-masters/password.txt.

1. You must create both files; they don't exist yet.

2. Do not use/modify the created files in the following steps, create new temporary files if needed.

Create a new secret names newsecret in the safe namespace, with the following content:

Username: dbadmin

Password: moresecurepas

Finally, create a new Pod that has access to the secret newsecret via a volume:

Namespace: safe

Pod name: mysecret-pod

Container name: db-container

Image: redis

Volume name: secret-vol

Mount path: /etc/mysecret

Answer: See the explanation below

Explanation:

```
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-cqqf6  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: QVU3dHh1bXFOTHZt
  username: cHVzHVjdGlvbi0x
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 "cHVzHVjdGlvbi0x" | base64 -d
production-lcandidate@cli:~$ 
candidate@cli:~$ 
candidate@cli:~$ echo "cHVzHVjdGlvbi0x" | base64 -d > /home/candidate/username.txt
candidate@cli:~$ cat /home/candidate/username.txt
production-lcandidate@cli:~$ 
candidate@cli:~$ 
candidate@cli:~$ echo "QVU3dHh1bXFOTHZt" | base64 -d
AU7txumqNLvmcandidate@cli:~$ echo "QVU3dHh1bXFOTHZt" | 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-data
base --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:~$ █
```

Question: 37

Cluster: scanner

Master node: controlplane

Worker node: worker1

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

[desk@cli] \$ kubectl config use-context scanner

Given:

You may use Trivy's documentation.

Task:

Use the Trivy open-source container scanner to detect images with severe vulnerabilities used by Pods in the namespace nato.

Look for images with High or Critical severity vulnerabilities and delete the Pods that use those images.

Trivy is pre-installed on the cluster's master node. Use cluster's master node to use Trivy.

Answer: See the explanation below

Explanation:

```
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 ls
```

```
-----+-----+
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@sha256:246ef631c75ea83005889621119fd5cc9ccb5500e1
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/cheewbacca
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 yaml
 3 for i in $(kubectl get pods -n naboo -o yaml); do kubectl get ${i} -o yaml | grep -i image ; done
 4 for i in $(kubectl get pods -n naboo -o yaml); 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 yaml); 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 yaml); 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 yaml); do kubectl -n naboo get ${i} -o yaml | grep -i image ; done
11 trivy image amazonlinux:2
12 kubectl get pods -n naboo -o yaml
13 kubectl -n naboo get pod/c-3po -o yaml | grep image
14 kubectl -n naboo delete pod/c-3po
15 kubectl -n naboo delete pod/jawas
16 kubectl -n naboo delete pod/jawas
17 history
root@kssc00401-master:~# []
```

Reference: <https://github.com/aquasecurity/trivy>

Question: 38

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

Answer: See the explanation below

Explanation:

```
master1 $ 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
$ vim netpol.yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
name: deny-network
namespace: test
spec:
```

```
podSelector: {}
policyTypes:
- Ingress
- Egress
master1 $ k apply -f netpol.yaml
```

Explanation

```
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
master1 $ k apply -f netpol1.yaml
```

Reference:

<https://kubernetes.io/docs/concepts/services-networking/network-policies/>

Explanation

```
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
master1 $ k apply -f netpol1.yaml
```

Reference:

<https://www.validexamdump.com/CKS.html>

<https://kubernetes.io/docs/concepts/services-networking/network-policies/>

Question: 39

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, runsc as well as traditional one.

Task:

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

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

Answer: See the explanation below

Explanation:

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

```
1  apiVersion: node.k8s.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
```

Explanation

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

```
apiVersion: node.k8s.io/v1
```

```
kind: RuntimeClass
```

```
metadata:
```

```
  name: not-trusted
```

```
handler: runsc
```

```
[desk@cli] $ k apply -f runtime.yaml
```

```
[desk@cli] $ k get pods
```

NAME	READY	STATUS	RESTARTS	AGE
nginx-6798fc88e8-chp6r	1/1	Running	0	11m
nginx-6798fc88e8-fs53n	1/1	Running	0	11m
nginx-6798fc88e8-ndved	1/1	Running	0	11m

```
[desk@cli] $ k get deploy
```

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
nginx	3/3	11	3	5m

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

```
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: {}
```

Reference: <https://kubernetes.io/docs/concepts/containers/runtime-class/>

Question: 40

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

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

Context:

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

Task:

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

1. Edit the existing Role bound to the Pod's ServiceAccount test-sa to only allow performing get operations, only on resources of type Pods.
2. Create a new Role named test-role-2 in the namespace database, which only allows performing update operations, only on resources of type statuefulsets.
3. Create a new RoleBinding named test-role-2-bind binding the newly created Role to the Pod's ServiceAccount.

Note: Don't delete the existing RoleBinding.

Answer: See the explanation below

Explanation:

```
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-8fdb-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:~$ 
```

Question: 41

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

Answer: See the explanation below

Explanation:

```
worker1 $ vim /var/lib/kubelet/config.yaml
```

anonymous:

```
enabled: true          #Delete this  
enabled: false         #Replace by this
```

authorization:

```
mode: AlwaysAllow      #Delete this  
mode: Webhook          #Replace by this
```

```
worker1 $ systemctl restart kubelet. # To reload kubelet config
```

ssh to master1

```
master1 $ vim /etc/kubernetes/manifests/kube-apiserver.yaml
```

```
-- authorization-mode=Node,RBAC
```

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

```
--client-cert-auth=true
```

Explanation

ssh to worker1

```
worker1 $ 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 config
```

ssh to master1

```
master1 $ 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
```

```
apiVersion: v1
kind: Pod
metadata:
  annotations:
    kubeadm.kubernetes.io/etcd.advertise-client-urls: https://172.17.0.29:2379
  creationTimestamp: null
  labels:
    component: etcd
    tier: control-plane
  name: etcd
  namespace: kube-system
spec:
  containers:
  - command:
    - etcd
    - --advertise-client-urls=https://172.17.0.29:2379
    - --cert-file=/etc/kubernetes/pki/etcd/server.crt
    - --client-cert-auth=true #Add this line
    - --data-dir=/var/lib/etcd
    - --initial-advertise-peer-urls=https://172.17.0.29:2380
    - --initial-cluster=controlplane=https://172.17.0.29:2380
    - --key-file=/etc/kubernetes/pki/etcd/server.key
    - --listen-client-urls=https://127.0.0.1:2379,https://172.17.0.29:2379
    - --listen-metrics-urls=http://127.0.0.1:2381
    - --listen-peer-urls=https://172.17.0.29:2380
    - --name=controlplane
    - --peer-cert-file=/etc/kubernetes/pki/etcd/peer.crt
    - --peer-client-cert-auth=true
    - --peer-key-file=/etc/kubernetes/pki/etcd/peer.key
    - --peer-trusted-ca-file=/etc/kubernetes/pki/etcd/ca.crt
    - --snapshot-count=10000
    - --trusted-ca-file=/etc/kubernetes/pki/etcd/ca.crt
  image: k8s.gcr.io/etcd:3.4.9-1
  imagePullPolicy: IfNotPresent
```

Reference:

kubelet parameters: <https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet/>

kubeapi parameters: <https://kubernetes.io/docs/reference/command-line-tools-reference/kube-apiserver/>

etcd parameters: <https://kubernetes.io/docs/tasks/administer-cluster/configure-upgrade-etcd/>

Question: 42

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



Cluster	Master node	Worker node
KSMV00	ksmv0030	
301	1-master	ksmv0030 1-worker1

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

```
[candidate@cli] $ | kubec  
tl config use-context KS  
MV00301
```

Context

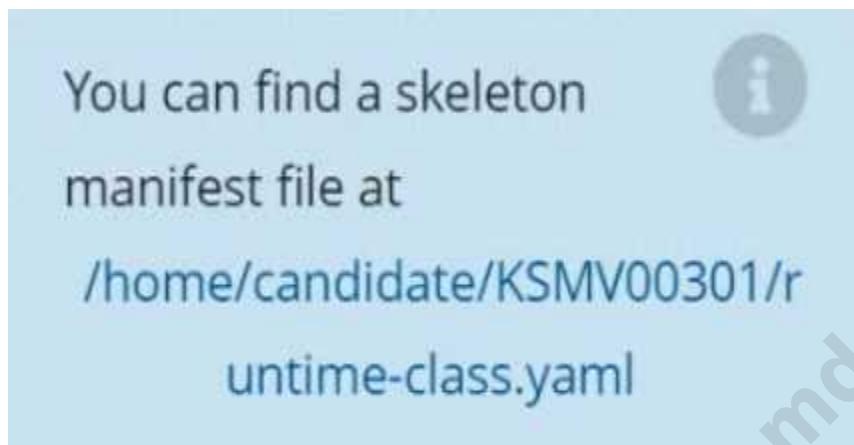
This cluster uses containerd as CRI runtime.

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.



Answer: See the explanation below

Explanation:

Explanation:

```
candidate@cli:~$ kubectl config use-context KSMV00301
Switched to context "KSMV00301".
candidate@cli:~$ cat /home/candidate/KSMV00301/runtime-class.yaml
---
apiVersion: node.k8s.io/v1
kind: RuntimeClass
metadata:
  name: ""
handler: ""
candidate@cli:~$ vim /home/candidate/KSMV00301/runtime-class.yaml
```

```
apiVersion: node.k8s.io/v1
kind: RuntimeClass
metadata:
  name: "sandboxed"
handler: "runsc"
~
```

:!ba:w!

```
candidate@cli:~$ kubectl config use-context KSMV00301
Switched to context "KSMV00301".
candidate@cli:~$ cat /home/candidate/KSMV00301/runtime-class.yaml
---
apiVersion: node.k8s.io/v1
kind: RuntimeClass
metadata:
  name: ""
handler: ""
candidate@cli:~$ vim /home/candidate/KSMV00301/runtime-class.yaml
candidate@cli:~$ cat /home/candidate/KSMV00301/runtime-class.yaml
---
apiVersion: node.k8s.io/v1
kind: RuntimeClass
metadata:
  name: "sandboxed"
handler: "runsc"
candidate@cli:~$ kubectl get deployments.apps -n server
NAME        READY   UP-TO-DATE   AVAILABLE   AGE
workload1   1/1     1           1           5h43m
workload2   1/1     1           1           5h43m
workload3   1/1     1           1           5h43m
candidate@cli:~$ kubectl get pods -n server
NAME                  READY   STATUS    RESTARTS   AGE
workload1-6869857dd7-s45rc   1/1     Running   0          5h43m
workload2-d4bd497d5-h44df   1/1     Running   0          5h43m
workload3-8587774495-chm56   1/1     Running   0          5h43m
candidate@cli:~$ kubectl -n server edit deployments.apps workload1
```

```
template:
  metadata:
    creationTimestamp: null
    labels:
      app: nginx
      name: workload1
  spec:
    runtimeClassName: sandboxed
    containers:
      - image: nginx:1.14.2
        imagePullPolicy: IfNotPresent
        name: workload1
        ports:
          - containerPort: 80
            protocol: TCP
        resources: {}
        terminationMessagePath: /dev/termination-log
        terminationMessagePolicy: File
    dnsPolicy: ClusterFirst
    restartPolicy: Always
    schedulerName: default-scheduler
    securityContext: {}
    terminationGracePeriodSeconds: 30
  status:
"/tmp/kubectl-edit-3385772700.yaml"
NAME           READY   STATUS    RESTARTS   AGE
workload1-6869857dd7-s45rc  1/1     Running   0          5h44m
workload2-d4bd497d5-h44df   1/1     Running   0          5h44m
workload3-8587774495-chm56  1/1     Running   0          5h44m
candidate@cli:~$ kubectl -n server edit deployments.apps workload1
Edit cancelled, no changes made.
candidate@cli:~$ kubectl get pods -n server
NAME           READY   STATUS    RESTARTS   AGE
workload1-6869857dd7-s45rc  1/1     Running   0          5h45m
workload2-d4bd497d5-h44df   1/1     Running   0          5h44m
workload3-8587774495-chm56  1/1     Running   0          5h44m
candidate@cli:~$ kubectl -n server edit deployments.apps workload2
Edit cancelled, no changes made.
candidate@cli:~$ kubectl create -f /home/candidate/KSMV00301/runtime-class.yaml
runtimeclass.node.k8s.io/sandboxed created
candidate@cli:~$ kubectl get pods -n server
NAME           READY   STATUS    RESTARTS   AGE
workload1-6869857dd7-s45rc  1/1     Running   0          5h45m
workload2-d4bd497d5-h44df   1/1     Running   0          5h45m
workload3-8587774495-chm56  1/1     Running   0          5h45m
candidate@cli:~$ kubectl -n server edit deployments.apps workload2
```

```
strategy:
  rollingUpdate:
    maxSurge: 25%
    maxUnavailable: 25%
  type: RollingUpdate
template:
  metadata:
    creationTimestamp: null
  labels:
    app: nginx
  name: workload2
spec:
  runtimeClassName: sandboxed
```

NAME	READY	STATUS	RESTARTS	AGE
workload1-6869857dd7-s45rc	1/1	Running	0	5h45m
workload2-d4bd497d5-h44df	1/1	Running	0	5h45m
workload3-8587774495-chm56	1/1	Running	0	5h45m

```
candidate@cli:~$ kubectl -n server edit deployments.apps workload2
deployment.apps/workload2 edited
candidate@cli:~$ kubectl get pods -n server
NAME                    READY   STATUS    RESTARTS   AGE
workload1-8d8649ff6-wvjtg   1/1    Running   0          15s
workload2-765bdb98c8-wd8cm   1/1    Running   0          4s
workload3-8587774495-chm56   1/1    Running   0          5h45m
candidate@cli:~$ kubectl -n server edit deployments.apps workload3
```

```
  app: nginx
  name: workload3
spec:
  runtimeClassName: sandboxed
  containers:
  - image: nginx:1.14.2
    imagePullPolicy: IfNotPresent
    name: workload3
  ports:
```

```
candidate@cli:~$ kubectl -n server edit deployments.apps workload3
deployment.apps/workload3 edited
candidate@cli:~$ kubectl get pods -n server
NAME                  READY   STATUS    RESTARTS   AGE
workload1-8d8649ff6-wvjtg   1/1     Running   0          58s
workload2-765bdb98c8-wd8cm   1/1     Running   0          47s
workload3-76c994bb4d-s6k85   1/1     Running   0          4s
candidate@cli:~$ █
```

Question: 43

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

Cluster	Master node	Worker node
KSCH00301	ksch00301 -master	ksch00301 -worker1

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

```
[candidate@cli] $ | kubectl config use-context KS CH00301
```

Context

Your organization's security policy includes:

ServiceAccounts must not automount API credentials

ServiceAccount names must end in "-sa"

The Pod specified in the manifest file /home/candidate/KSCH00301/pod-manifest.yaml fails to schedule because of an incorrectly specified ServiceAccount.

Complete the following tasks:

Task

1. Create a new ServiceAccount named frontend-sa in the existing namespace qa

a. Ensure the ServiceAccount does not automount API credentials.

2. Using the manifest file at /home/candidate/KSCH00301/pod-manifest.yaml, create the Pod.

3. Finally, clean up any unused ServiceAccounts in namespace qa.

Answer: See the explanation below

Explanation:

```
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:~$ []
```

Question: 44

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

Cluster	Master node	Worker node
KSCS00201	kscs00201 -master	kscs00201 -worker1

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

```
[candidate@cli] $ | kubec  
tl config use-context KS  
CS00201
```

Context

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

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:

Ensure that the
--authorization

1.2.7 -mode FAIL
argument is not
set to

AlwaysAllow

Ensure that the
--authorization

1.2.8 -mode FAIL
argument
includes Node

Ensure that the
--authorization

1.2.9 -mode FAIL
argument
includes RBAC

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

Ensure that the
anonymous-au



4.2.1 th FAIL
argument is set
to false

Ensure that the
--authorization

4.2.2 -mode FAIL
argument is not
set to
AlwaysAllow

Use Webhook
authentication/authorization
where possible.



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

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

Answer: See
explanation below.

Explanation:

```
candidate@cli:~$ kubectl delete sa/podrunner -n qa
serviceaccount "podrunner" deleted
candidate@cli:~$ kubectl config use-context KSCS00201
Switched to context "KSCS00201".
candidate@cli:~$ ssh ksks00201-master
Warning: Permanently added '10.240.86.194' (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@kscs00201-master:~# vim /etc/kubernetes/manifests/kube-apiserver.yaml
root@kscs00201-master:~# systemctl daemon-reload
root@kscs00201-master:~# systemctl restart kubelet.service
root@kscs00201-master:~# systemctl enable kubelet.service
root@kscs00201-master:~# systemctl status kubelet.service
● kubelet.service - kubelet: The Kubernetes Node Agent
   Loaded: loaded (/lib/systemd/system/kubelet.service; enabled; vendor preset: enabled)
   Drop-In: /etc/systemd/system/kubelet.service.d
             └─10-kubeadm.conf
     Active: active (running) since Fri 2022-05-20 14:19:31 UTC; 29s ago
       Docs: https://kubernetes.io/docs/home/
   Main PID: 134205 (kubelet)
     Tasks: 16 (limit: 76200)
    Memory: 39.5M
      CGroup: /system.slice/kubelet.service
              └─134205 /usr/bin/kubelet --bootstrap-kubeconfig=/etc/kubernetes/bootstrap-kub>
```

```
May 20 14:19:35 ksks00201-master kubelet[134205]: I0520 14:19:35.420825 134205 reconciler.>
May 20 14:19:35 ksks00201-master kubelet[134205]: I0520 14:19:35.420863 134205 reconciler.>
May 20 14:19:35 ksks00201-master kubelet[134205]: I0520 14:19:35.420907 134205 reconciler.>
May 20 14:19:35 ksks00201-master kubelet[134205]: I0520 14:19:35.420928 134205 reconciler.>
May 20 14:19:36 ksks00201-master kubelet[134205]: I0520 14:19:36.572353 134205 request.go:>
May 20 14:19:37 ksks00201-master kubelet[134205]: I0520 14:19:37.112347 134205 prober_mana>
May 20 14:19:37 ksks00201-master kubelet[134205]: E0520 14:19:37.185076 134205 kubelet.go:>
May 20 14:19:37 ksks00201-master kubelet[134205]: I0520 14:19:37.645798 134205 kubelet.go:>
May 20 14:19:38 ksks00201-master kubelet[134205]: I0520 14:19:38.184062 134205 kubelet.go:>
May 20 14:19:40 ksks00201-master kubelet[134205]: I0520 14:19:40.036042 134205 prober_mana>
lines 1-22/22 (END)
```

```
de Agent
et.service; enabled; vendor preset: enabled)
ce.d

5-20 14:19:31 UTC; 29s ago

trap-kubeconfig=/etc/kubernetes/bootstrap-kubelet.conf --kubeconfig=/etc/kubernetes/kubelet>
5]: I0520 14:19:35.420825 134205 reconciler.go:221] "operationExecutor.VerifyControllerAtt>
5]: I0520 14:19:35.420863 134205 reconciler.go:221] "operationExecutor.VerifyControllerAtt>
5]: I0520 14:19:35.420907 134205 reconciler.go:221] "operationExecutor.VerifyControllerAtt>
5]: I0520 14:19:35.420928 134205 reconciler.go:157] "Reconciler: start to sync state"
5]: I0520 14:19:36.572353 134205 request.go:665] Waited for 1.049946364s due to client-sid>
5]: I0520 14:19:37.112347 134205 prober_manager.go:255] "Failed to trigger a manual run" p>
5]: E0520 14:19:37.185076 134205 kubelet.go:1711] "Failed creating a mirror pod for" err=">
5]: I0520 14:19:37.645798 134205 kubelet.go:1693] "Trying to delete pod" pod="kube-system/>
5]: I0520 14:19:38.184062 134205 kubelet.go:1698] "Deleted mirror pod because it is outdated"
5]: I0520 14:19:40.036042 134205 prober_manager.go:255] "Failed to trigger a manual run" p>
~
~
lines 1-22/22 (END)

let.conf --kubeconfig=/etc/kubernetes/kubelet.conf --config=/var/lib/kubelet/config.yaml -->
o:221] "operationExecutor.VerifyControllerAttachedVolume started for volume \"kube-proxy\" >
o:221] "operationExecutor.VerifyControllerAttachedVolume started for volume \"lib-modules\" >
o:221] "operationExecutor.VerifyControllerAttachedVolume started for volume \"flannel-cfg\" >
o:157] "Reconciler: start to sync state"
65] Waited for 1.049946364s due to client-side throttling, not priority and fairness, reques>
er.go:255] "Failed to trigger a manual run" probe="Readiness"
711] "Failed creating a mirror pod for" err="pods \"kube-apiserver-kscs00201-master\" alrea>
693] "Trying to delete pod" pod="kube-system/kube-apiserver-kscs00201-master" podUID=bb91e1>
698] "Deleted mirror pod because it is outdated" pod="kube-system/kube-apiserver-kscs00201->
er.go:255] "Failed to trigger a manual run" probe="Readiness"
~
~
root@kscs00201-master:~# vim /var/lib/kubelet/config.yaml
```

```
apiVersion: kubelet.config.k8s.io/v1beta1
authentication:
  anonymous:
    enabled: false
  webhook:
    cacheTTL: 0s
    enabled: true
  x509:
    clientCAFile: /etc/kubernetes/pki/ca.crt
authorization:
  mode: Webhook
  webhook:
    cacheAuthorizedTTL: 0s
    cacheUnauthorizedTTL: 0s
cgroupDriver: systemd
clusterDNS:
~
~
```

root@kscs00201-master:~# vim /var/lib/kubelet/config.yaml
root@kscs00201-master:~# vim /var/lib/kubelet/config.yaml
root@kscs00201-master:~# vim /etc/kubernetes/manifests/etcd.yaml
root@kscs00201-master:~# systemctl daemon-reload
root@kscs00201-master:~# systemctl restart kubelet.service
root@kscs00201-master:~# systemctl status kubelet.service

- **kubelet.service - kubelet: The Kubernetes Node Agent**
 Loaded: loaded (/lib/systemd/system/kubelet.service; enabled; vendor preset: enabled)
 Drop-In: /etc/systemd/system/kubelet.service.d
 └─10-kubeadm.conf
 Active: active (running) since Fri 2022-05-20 14:22:29 UTC; 4s ago
 Docs: https://kubernetes.io/docs/home/
 Main PID: 135849 (kubelet)
 Tasks: 17 (limit: 76200)
 Memory: 38.0M
 CGroup: /system.slice/kubelet.service
 └─135849 /usr/bin/kubelet --bootstrap-kubeconfig=/etc/kubernetes/bootstrap-kub>

May 20 14:22:30 kscs00201-master kubelet[135849]: I0520 14:22:30.330232 135849 reconciler.>
May 20 14:22:30 kscs00201-master kubelet[135849]: I0520 14:22:30.330259 135849 reconciler.>
May 20 14:22:30 kscs00201-master kubelet[135849]: I0520 14:22:30.330304 135849 reconciler.>
May 20 14:22:30 kscs00201-master kubelet[135849]: I0520 14:22:30.330354 135849 reconciler.>
May 20 14:22:30 kscs00201-master kubelet[135849]: I0520 14:22:30.330378 135849 reconciler.>
May 20 14:22:30 kscs00201-master kubelet[135849]: I0520 14:22:30.330397 135849 reconciler.>
May 20 14:22:30 kscs00201-master kubelet[135849]: I0520 14:22:30.330415 135849 reconciler.>
May 20 14:22:30 kscs00201-master kubelet[135849]: I0520 14:22:30.330433 135849 reconciler.>
May 20 14:22:30 kscs00201-master kubelet[135849]: I0520 14:22:30.330452 135849 reconciler.>
May 20 14:22:30 kscs00201-master kubelet[135849]: I0520 14:22:30.330463 135849 reconciler.>
- lines 1-22/22 (END)

```
May 20 14:22:30 ksks00201-master kubelet[135849]: I0520 14:22:30.330463 135849 reconciler.>
root@ksks00201-master:~#
root@ksks00201-master:~#
root@ksks00201-master:~#
root@ksks00201-master:~# exit
logout
Connection to 10.240.86.194 closed.
candidate@cli:~$
```

Question: 45

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

Cluster	Master node	Worker node
KSCS001 01	kscs00101 -master	kscs00101 -worker1

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

```
[candidate@cli] $ | kubec
t1 config use-context KS
CS00101
```

Context

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/n
etwork-policy.yaml`

**Answer: See
explanation below.**

Explanation:

```
candidate@cli:~$ kubectl config use-context KSCS00101
Switched to context "KSCS00101".
candidate@cli:~$ cat /home/candidate/KSCS00101/network-policy.yaml
---
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: ""
  namespace: ""
spec:
  podSelector: {}
  policyTypes: []
candidate@cli:~$ vim /home/candidate/KSCS00101/network-policy.yaml
candidate@cli:~$ █
```

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: "defaultdeny"
  namespace: "testing"
spec:
  podSelector: {}
  policyTypes:
    - Egress
  egress:
    - to:
        - podSelector: {}
          namespaceSelector:
            matchLabels:
              access: testingproject
```

```
candidate@cli:~$ vim /home/candidate/KSCS00101/network-policy.yaml
candidate@cli:~$ vim /home/candidate/KSCS00101/network-policy.yaml
candidate@cli:~$ kubectl label ns testing access=testingproject
namespace/testing labeled
candidate@cli:~$ cat /home/candidate/KSCS00101/network-policy.yaml
---
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: "defaultdeny"
  namespace: "testing"
spec:
  podSelector: {}
  policyTypes:
  - Egress
  egress:
  - to:
    - podSelector: {}
      namespaceSelector:
        matchLabels:
          access: testingproject
candidate@cli:~$ kubectl create -f /home/candidate/KSCS00101/network-policy.yaml
networkpolicy.networking.k8s.io/defaultdeny created
candidate@cli:~$ kubectl -n testing describe networkpolicy
Name:           defaultdeny
Namespace:      testing
Created on:     2022-05-20 14:28:27 +0000 UTC
Labels:         <none>
Annotations:   <none>
Spec:
  PodSelector:    <none> (Allowing the specific traffic to all pods in this namespace)
  Not affecting ingress traffic
  Allowing egress traffic:
    To Port: <any> (traffic allowed to all ports)
    To:
      NamespaceSelector: access=testingproject
      PodSelector: <none>
    Policy Types: Egress
candidate@cli:~$ █
```

Question: 46

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

```
[candidate@cli] $ | kubec  
tl config use-context KS  
MV00102
```

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.

You can find skeleton
manifest files at:

- /home/candidate/KSMV00
102/pod-security-policy.ya
ml
- /home/candidate/KSMV00
102/cluster-role.yaml
- /home/candidate/KSMV00
102/service-account.yaml
- /home/candidate/KSMV00
102/cluster-role-binding.ya
ml

**Answer: See
explanation below.**

Explanation:

```
candidate@cli:~$ kubectl config use-context KSMV00102
Switched to context "KSMV00102".
candidate@cli:~$ cat /home/candidate/KSMV00102/pod-security-policy.yaml
---
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  name: ""
spec:
  seLinux:
    rule: ""
  runAsUser:
    rule: ""
  supplementalGroups: {}
  fsGroup: {}
candidate@cli:~$ vim /home/candidate/KSMV00102/pod-security-policy.yaml
```

```
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  name: "prevent-psp-policy"
spec:
  privileged: false
  seLinux:
    rule: RunAsAny
  runAsUser:
    rule: RunAsAny
  supplementalGroups:
    rule: RunAsAny
  fsGroup:
    rule: RunAsAny
```

```
candidate@cli:~$ vim /home/candidate/KSMV00102/pod-security-policy.yaml
candidate@cli:~$ cat /home/candidate/KSMV00102/pod-security-policy.yaml
---
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  name: "prevent-psp-policy"
spec:
  privileged: false
  seLinux:
    rule: RunAsAny
  runAsUser:
    rule: RunAsAny
  supplementalGroups:
    rule: RunAsAny
  fsGroup:
    rule: RunAsAny
candidate@cli:~$ kubectl create -f /home/candidate/KSMV00102/pod-security-policy.yaml
Warning: policy/v1beta1 PodSecurityPolicy is deprecated in v1.21+, unavailable in v1.25+
podsecuritypolicy.policy/prevent-psp-policy created
candidate@cli:~$ cat /home/candidate/KSMV00102/cluster-role.yaml
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: ""
rules:
candidate@cli:~$ vim /home/candidate/KSMV00102/cluster-role.yaml
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: "restrict-access-role"
rules:
```

```
candidate@cli:~$ kubectl create clusterrole restrict-access-role --verb=use --resource=psp \
--dry-run=client -o yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  creationTimestamp: null
  name: restrict-access-role
rules:
- apiGroups:
  - policy
  resources:
  - podsecuritypolicies
  verbs:
  - use
candidate@cli:~$ vim /home/candidate/KSMV00102/cluster-role.yaml
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: "restrict-access-role"
rules:
- apiGroups:
  - policy
  resources:
  - podsecuritypolicies
  verbs:
  - use
candidate@cli:~$ vim /home/candidate/KSMV00102/cluster-role.yaml
candidate@cli:~$ kubectl create clusterrole restrict-access-role --verb=use --resource=psp -
-dry-run=client --resource-name=prevent-psp-policy -o yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  creationTimestamp: null
  name: restrict-access-role
rules:
- apiGroups:
  - policy
  resourceNames:
  - prevent-psp-policy
  resources:
  - podsecuritypolicies
  verbs:
  - use
candidate@cli:~$ vim /home/candidate/KSMV00102/cluster-role.yaml
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: "restrict-access-role"
rules:
- apiGroups:
  - policy
  resourceNames:
  - prevent-psp-policy
  resources:
  - podsecuritypolicies
  verbs:
  - use
```

```
candidate@cli:~$ kubectl create -f /home/candidate/KSMV00102/cluster-role.yaml
clusterrole.rbac.authorization.k8s.io/restrict-access-role created
candidate@cli:~$
candidate@cli:~$
candidate@cli:~$ cat /home/candidate/KSMV00102/service-account.yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: "psp-restrict-sa"
  namespace: "staging"
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: ""
  namespace: ""
candidate@cli:~$ vim /home/candidate/KSMV00102/service-account.yaml
candidate@cli:~$ cat /home/candidate/KSMV00102/service-account.yaml
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: "psp-restrict-sa"
  namespace: "staging"
candidate@cli:~$ kubectl get sa -n staging
NAME      SECRETS   AGE
default    1          6h6m
candidate@cli:~$ kubectl create -f /home/candidate/KSMV00102/service-account.yaml
serviceaccount/psp-restrict-sa created
candidate@cli:~$ kubectl get sa -n staging
NAME      SECRETS   AGE
default    1          6h6m
psp-restrict-sa  1          2s
candidate@cli:~$ 
candidate@cli:~$ kubectl create clusterrolebinding restrict-access-bind --clusterrole=restrict-access-role --serviceaccount=staging:psp-restrict-sa --dry-run -o yaml
W0520 14:41:23.502004    47627 helpers.go:598] --dry-run is deprecated and can be replaced with --dry-run=client.
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  creationTimestamp: null
  name: restrict-access-bind
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: restrict-access-role
subjects:
- kind: ServiceAccount
  name: psp-restrict-sa
  namespace: staging
candidate@cli:~$ vim /home/candidate/KSMV00102/cluster-role-binding.yaml
cluster-role-binding.yaml  cluster-role.yaml
candidate@cli:~$ vim /home/candidate/KSMV00102/cluster-role-binding.yaml
cluster-role-binding.yaml  cluster-role.yaml
candidate@cli:~$ vim /home/candidate/KSMV00102/cluster-role-binding.yaml
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: restrict-access-bind
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: restrict-access-role
subjects:
- kind: ServiceAccount
  name: psp-restrict-sa
  namespace: staging

apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: restrict-access-bind
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: restrict-access-role
subjects:
- kind: ServiceAccount
  name: psp-restrict-sa
  namespace: staging

candidate@cli:~$ kubectl create -f /home/candidate/KSMV00102/cluster-role-binding.yaml
clusterrolebinding.rbac.authorization.k8s.io/restrict-access-bind created
candidate@cli:~$ 
```

Question: 47

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] $ | kubec  
tl config use-context K  
SC00301
```

Task

Analyze and edit the given Dockerfile /home/candidate/KSSC00301/Dockerfile (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.

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`.

Answer: See explanation below.

Explanation:

```
candidate@cli:~$ kubectl config use-context KSSC00301
Switched to context "KSSC00301".
candidate@cli:~$ vim KSSC00301/Dockerfile
```

```

FROM ubuntu:16.04

USER root
RUN apt-get update && \
    apt-get install -yq --no-install-recommends runit=2.1.2-3ubuntu1 wget=1.17.1-1ubuntu1.5 \
    \ chrpath=0.16-1 tzdata=2020a-0ubuntu0.16.04 lsof=4.89+dfsg-0.1 lshw=02.17-1.1ubuntu3 \
    \ sysstat=11.2.0-1ubuntu0.3 net-tools=1.60-26ubuntu1 numactl=2.0.11-1ubuntu1.1 \
    bzip2=1.0.6-8ubuntu0.2 && \
    apt-get autoremove && apt-get clean && \
    rm -rf /var/lib/apt/lists/* /tmp/* /var/tmp/*

ARG CB_VERSION=6.5.1
ARG CB_RELEASE_URL=https://packages.couchbase.com/releases/6.5.1
ARG CB_PACKAGE=couchbase-server-enterprise_6.5.1-ubuntu16.04_amd64.deb
ARG CB_SHA256=80427193137e5cb5a4795b2675b1c450clf8cf1a5c634d917f6c416f2047e66

ENV PATH=$PATH:/opt/couchbase/bin:/opt/couchbase/bin/tools:/opt/couchbase/bin/install

RUN groupadd -g 1000 couchbase && useradd couchbase -u 1000 -g couchbase -M

SHELL ["/bin/bash", "-o", "pipefail", "-c"]
RUN export INSTALL_DONT_START_SERVER=1 && \
    wget -N --no-verbose $CB_RELEASE_URL/$CB_PACKAGE && \
    echo "$CB_SHA256 $CB_PACKAGE" | sha256sum -c - && \
    dpkg -i ./${CB_PACKAGE} && rm -f ./${CB_PACKAGE}

COPY scripts/run /etc/service/couchbase-server/run
RUN chown -R couchbase:couchbase /etc/service

COPY scripts/dummy.sh /usr/local/bin/
RUN ln -s dummy.sh /usr/local/bin/iptables-save && \
    ln -s dummy.sh /usr/local/bin/lvdisplay && \
    ln -s dummy.sh /usr/local/bin/vgdisplay && \
    ln -s dummy.sh /usr/local/bin/pvdisplay

RUN chrpath -r "$ORIGIN/../lib" /opt/couchbase/bin/curl

COPY scripts/entrypoint.sh /
ENTRYPOINT ["/entrypoint.sh"]
USER nobody
CMD ["couchbase-server"]

EXPOSE 8091 8092 8093 8094 8095 8096 11207 11210 11211 18091 18092 18093 18094 18095 18096
VOLUME /opt/couchbase/var

```

candidate@cli:~\$ kubectl config use-context KSSC00301
Switched to context "KSSC00301".
candidate@cli:~\$ vim KSSC00301/Dockerfile
candidate@cli:~\$ vim KSSC00301/deployment.yaml

```

    securityContext:
      capabilities: {'add': ['NET_BIND_SERVICE'], 'drop': ['all']}, 'privileged': False, 'readOnlyRootFilesystem': True, 'runAsUser': 65535
      resources:
        limits:
          cpu: 2
          memory: 1024Mi
        requests:
          cpu: 1
          memory: 512Mi
      volumes:
        - name: database-storage

```

Question: 48

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
```

Context

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

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.

Answer: See

explanation below.

Explanation:

```
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
Z0lCQURBTkJna3Foa2lHOXcwQkFRc0ZBREFWTVJNd0VRWURWUVFERXdwcmRXSmwKY201bGRHVnpNQjRYRFRJeU1ESXho
akF3T1RVeE9Wb1hEVE15TURJeE5EQXdOVFV4T1zvd0ZURVRNQkVHQTFRQpBeE1LYTNWaVpYSnVaWFJsY3pDQ0FTSxdE
UV1KS29aSwH2Y05BUUVCQ1FBRGdnRVBBRENDQFvQ2dnRUJBTlgwCm9LeUYvTGNmYT1vNzNTktksFdZU3JUaUx0QStr
N01gTxpRZllzM2ttNGllalp0tZc3Y1bUdpN0UyQ2tYc0MKUh1lNiZnBDMz1la2k5V3hOSHc5eTM0OEtXUVE3VXBL
UmZRdxVxd1A1WxdDZkord1JmWGNGTXQxLzRNQVhWLwpkjZ5YWRKSlPeFFSVjZlaHFBZHR0M3FtOFdVcW84UE5JT1E0
OEc3WWhnRUg5RHU3SFdkMS8raXvkSjNOMX16CnNISEdtYklisWENsbEcydFV0M2RScDczSnRIS1Jjs2tnMGxYM3FWS1Uy
QmJRblBmK01wb0VlTxFGcmZvcWVaWcKY1BKK3ROVmZIM1JLTkhVUnYdVJ1a3Zzc2JrclhUMW8rMXFNNHZrYnFNMH1q
KzNxTutiSy5V3dzU1tBYUVPMPapUdXR4Ud1TFp3OUE3TjZZeTFVQ0F3RUFByU5aTUZjd0RnWURWUjBQQVFIL0JBURB
Z0trTUE4ROExVWRFd0VCC193UUZQNQ1CQWY4d0hRWURWUjBPQKJZRUZEcUlwlZdYbzZaNkNVjVEK2w3bFZPcGpBOW1N
Q1VHQTFVZEVRU8KTUF5Q0ntdDFzbVZ5Ym1WMFpYTXdEUV1KS29aSwH2Y05BUUVMQ1FBRGdnRUJES1NWNm9wNGgxYkNv
eGZLRUZ4bwxaV1HUF1nM1hHTN0WEZ1TTY3RnA2NkdqUEc5SXBONhNUhRnWV1yd0MyalBDfFVOb2IySWtUQ1FNbDV3
cWRHCkdPS2JwVvp6Smc3Y0dyS2E3R1pZWVNyUVVGRWhyd2xZWXNGME56aFBoZCwcHJjcWtSdXN1bm55SG5YNGVOMUoK
N1NzbGZYTjJ1dVfJd1VIRG15L0JsL1ZWRmZNzRxOGF0Z0pYSFZG7m1vCDRpNX1JTXFRNTB4ZjVqcnF1WFRmVwpVdmJq
ZjEyOThXVTk3QkxHcDdRZE9QyWVKU051Us1VkmRdnpV2zTQVNjclVsc24xcThPnnBRbjV3TjNxdUVrCm5zQk9pckxs
c2k2a1N3UlhLbGevangvcitqd0Tc0xwWUxDZTlxalFraTdCSVRJT1N3ejd3c2hzbERuNzBFY0IKa0VBPQotLS0tLUVO
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: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tck1JSURJVENDQWdtZ0F3SUJBZ0l
ocEdQcDB4Zk9JbkYxaGJwcTh5Y1BUMGx1Tm5VNjBiSupxRXVKckxBtEtXC1NValh1VkyZnk10ZhclZU10T2JxK1haaHd
hY2JURVZCM1VDVURsbDgzdG5teFQyVXJmY0pUQmhLTCTzTFAvcWYKdjdxR3BwQ1ZXNnhVZGfibGnuUk1IMnpleUVJTEt
Tck5XbUQ0TzZsMU13b1Z00VjzQ2RXTkV3VGNZRHdoUTd20QpGcExKL3hiSDdUTZkwY1Rfd1Iwazl3cFVYd11kdk1jSXN
MRKyWL3F2bDA3U31xbGp10EI1SnNpQ1hCU1ZxbS9wCmNUUSS3Sn1bmndaZz1kOWdzaVJVdFFTchBONkx4UnhkSzNKMGR
BK240SWxFZEtHRWh3TE00d0tMaldeRG9scHgKYzB3WHkwVBORGZ6UxuRUFzVUJsbDRCQ3VkdW5QNVVDN2FuS3dJREF
RQUJb01CQURWRkZNSVRqYnNySTZTtWpQOGM0MTBynJRWZ251cXJVS202dHrn2WtX0wd1S1pvMnZyb3RsbG9qOGFRamF
0MTZnaEUwOXd2d2xMSDhId0tLcL1Mb2NrZnFCUyt10w1Zm1FWGxyTG00cE1CVDFRbGFJQ1JRMDRyQ0JZbhdCN1VFbVB
1Wjhuc31mR2JYTC9HM2wKcXBTDVkdzJqcVh2MXdzcWsrdWNCRk0z20FYzkh1RExnV0VyNKRZr1F4Vxo5UFFHOD1
pcDY10TBkYnB1SApOMnU2NGk4UTg1dk830FVIT1c2eUFZU1loZvdha093RFwzNPdkhxV3FhbnV1MnlrOWxaUUR0W5
2MytBeU5DChn0N1RaRhlz01ZdEphtDFtQ01TNEpSRZd4NXNwaCtKOC9XOGx0R19wMzxbTA0bXZSRndxU3M2Y1JCQ2Z
PVVcKbFV1MGxLRUNnWUVBNWJzT01VVzFBVndjTmJscOpSVDNURk1Z0V1xbDRYcnZRR0FZY3BhdktENnd5VmtEOTV1QOp
SaXVRs1NNKzY4REtBVmlpY1lpaThJemExTkdqdC9J2DUwTGVoNk1aRVg2enVpk0g3d1BSbVd6SE9ueWNmu2FmC1VQMEF
RL0RiM21CNWJQ7mJHYXNkaDNib2JvR0NSHmTFFXY2tYbUVXM2ZudV1IR1JLZ2x1TEVD211FQTVUdysKTEVTv1BESFF
manNBN0htNmDsMndGRjdCUG1ssGdaYVVRN25Eb3ZvRmxa1BMRWVCMWJ6OHJNW1d1eGdmaHN0OqpMZ0xSUDBXdkJwd1J
sVTdMTmFLT1VzRmkxU2dvaWzsS01ZwKMyZmpLWTY1RF3YUUzctDnVis4U2pIZHpoclhCCKVQclAvWXQ3S0QrbFBMZh
aNKNZWftelY3b3gveno5Y0s0U0ZKc0NnWUJ40Vk2v2FydhB0MfcvS05JS3V4SeoKmjRxRFQxbm1ObE9FdmFhakFUatTJ
ZQkxXYnIvWERsNWRjTEs4bfCxYkNYR3JwY2s3U0xKn1hZUV1XajQ2dTNJMgpEQ2ZUW1f1RWRQzNBBwtPR2ZqWmdPcDd
pdUVkL0JDLzNpRkprcXV1enNFdFdMTH1Vcjm5T0hZeW16QVJ4Tmo2CnzuUG1ma000Rk16d3F2MHVoN0x1b1FIQmdBT1Z
XZTRZM1RwbzJ3aEswbmVm11sMxhvnjJoZ2JiVHcvaVdhVcKY3ZMV3dlZU1md0Q4MVRWL2R3a29KVEM1VEJRXQzUkk
xRFFnVmtmMFHFcwUtoOGhDNGQwM05MRzIwTwdZMk94WgpjSFZzK2J4e1YwVVB6V1RUbEmyVesyamhmOHVRcndzSktxY2N
OU02EcZwclhocThsV213Znd3aW1BRlhLSFJRCkE3RkxBb0dCQx3NW8rbHFVZ3hHQlpKdy9Ee1RGek5TekQreVd6Um8
1c22Ec2x6a2FvY0pHbEx2MUNndEVic3QKeG5HMT1IYStSM1M3cDRtei9LeDJYMFRAzATzzUzVwW1R5WEs5STF5azh2TUZ
rRldacjRmeVhXV2t3Sjz1VE11YwpyWF13TWM5VFlDUGzrSFJaTm9XR1hZV3BkeTJBOXZCbFlSchZsQVZoenU2T1VZQ2w
5b2ZpcI0tLS0tRU5E1FJTQSBQuk1WQVRF1etFWS0tLS0tCg==
root@ksch00101-master:~# vim /etc/kubernetes/manifests/kube-apiserver.yaml

```

```
apiVersion: v1
kind: Pod
metadata:
  annotations:
    kubernetes.io/kube-apiserver-advertise-address.endpoint: 10.240.86.190:6443
  creationTimestamp: "2021-01-12T10:44:11Z"
  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/etcd-client.crt
    - --etcd-keyfile=/etc/kubernetes/pki/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-user-name-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: 3
    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
    initialDelaySeconds: 10
    periodSeconds: 10
    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
    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
```

```
root@ksch00101-master:~# vim /etc/kubernetes/manifests/kube-apiserver.yaml
root@ksch00101-master:~# systemctl daemon-reload
root@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:anon
us
clusterrolebinding.rbac.authorization.k8s.io "system:anonymous" deleted
```

Thank you for your visit.

To try more exams, please visit below link
<https://www.validexamdumps.com/CKS.html>