CKS Exam Questions and Answers

Question 1: Falco Runtime Security Detection

Weight: 7% | **Domain:** Monitoring, Logging & Runtime Security | **Difficulty:** Medium

Question

Three deployments exist in namespace `apps`: `nvidia-gpu`, `cpu`, and `ollama`. Pods from one of these deployments are accessing `/dev/mem`, causing memory issues on the node.

Task:

1. Use Falco to identify which pod is accessing `/dev/mem`
2. Scale the related deployment to 0 replicas
3. Save findings:
    - Pod name to: `/opt/course/01/pod-name.txt`
    - Falco alert line to: `/opt/course/01/falco-alert.txt`

Answer

```
# Find which node the apps pods are running on
kubectl get pods -n apps -o wide

# SSH to that node and check Falco logs
ssh <node-name> journalctl -u falco --no-pager | grep -i mem

# Scale down the offending deployment (ollama)
kubectl scale deployment ollama -n apps --replicas=0

# Save findings
mkdir -p /opt/course/01
kubectl get pods -n apps -l app=ollama -o
jsonpath='{.items[0].metadata.name}' > /opt/course/01/pod-name.txt
echo "<timestamp> Notice Read sensitive device /dev/mem by
container=ollama pod=ollama-xxxxx" > /opt/course/01/falco-alert.txt
```

Question 2: Worker Node Kubernetes Upgrade

Weight: 5% | **Domain:** Cluster Hardening | **Difficulty:** Medium

Question

A worker node named `node-01` is running kubelet version 1.34.0. The control plane is already at version 1.34.1.

Task: Upgrade the worker node to version 1.34.1

Save:

- Pre-upgrade version: `/opt/course/02/node-version-before.txt`
- Post-upgrade version: `/opt/course/02/node-version-after.txt`

Answer

```
# Save current version (from controlplane)
kubectl get nodes <worker-node> -o jsonpath='{.status.nodeInfo.kubeletVersion}' > /opt/course/02/node-version-before.txt

# SSH to worker node
ssh <worker-node>

# Update apt and check the kubeadm package
sudo apt update
sudo apt-cache madison kubeadm

# Check the current kubeadm package afterwards
sudo dpkg -l | grep kubeadm

# Upgrade kubeadm
sudo apt-mark unhold kubeadm && \
sudo apt-get update && sudo apt-get install -y kubeadm='1.34.1-1.1' && \
sudo apt-mark hold kubeadm

# Upgrade node configuration
sudo kubeadm upgrade node

# From controlplane (new terminal) - drain the node
kubectl drain <node-name> --ignore-daemonsets --delete-emptydir-data --force

# Back on worker node - upgrade kubelet and kubectl
sudo apt-mark unhold kubelet kubectl && \
sudo apt-get update && sudo apt-get install -y kubelet='1.34.1-1.1' kubectl='1.34.1-1.1' && \
sudo apt-mark hold kubelet kubectl

# Restart kubelet
sudo systemctl daemon-reload
sudo systemctl restart kubelet
exit

# From controlplane - uncordon
kubectl uncordon <worker-node>

# Save post-upgrade version
kubectl get nodes <worker-node> -o jsonpath='{.status.nodeInfo.kubeletVersion}' > /opt/course/02/node-version-after.txt
```

Question 3: Ingress with TLS

Weight: 5% | **Domain:** Cluster Setup | **Difficulty:** Easy

Question

A TLS secret named `tls-secret` exists in namespace `secure-app`. A service named `secure-service` is running on port 80.

Task: Create an Ingress named `secure-ingress` that:

- Uses TLS secret `tls-secret`
- Redirects HTTP to HTTPS
- Routes traffic for `secure.example.com` to `secure-service:80`

Save: `/opt/course/03/ingress.yaml`

Answer

```
# Check the ingress-nginx controller is installed and running
k -n ingress-nginx get pods

# Generate the Ingress manifest using kubectl
kubectl -n secure-app create ingress secure-ingress \
    --class=nginx \
    --rule="secure.example.com/*=secure-service:80,tls=tls-secret" \
    --annotation="nginx.ingress.kubernetes.io/ssl-redirect=true" \
    --dry-run=client -o yaml | tee /opt/course/03/ingress.yaml

# Apply it
kubectl apply -f /opt/course/03/ingress.yaml
```

Or manually create the YAML:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: secure-ingress
  namespace: secure-app
  annotations:
    nginx.ingress.kubernetes.io/ssl-redirect: "true"
spec:
  ingressClassName: nginx
  tls:
  - hosts:
    - secure.example.com
    secretName: tls-secret
  rules:
  - host: secure.example.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: secure-service
            port:
              number: 80
```

```
kubectl apply -f /opt/course/03/ingress.yaml
```

Question 4: SBOM Generation

Weight: 4% | **Domain:** Supply Chain Security | **Difficulty:** Easy

Question

Analyze the software components inside a container image for supply chain security purposes using multiple SBOM tools.

Image: `nginx:1.25-alpine@sha256:721fa00bc549df26b3e67cc558ff176112d4ba69847537766f3c28e171d180e7`

Task:

1. Use `bom` to generate an SBOM in SPDX format: `/opt/course/04/sbom.spdx`
2. Use `trivy` to generate an SBOM in SPDX-JSON format: `/opt/course/04/sbom.spdx.json`
3. Use `trivy` to scan the SBOM for vulnerabilities: `/opt/course/04/sbom-vulns.json`
4. Query for SSL packages, save to: `/opt/course/04/ssl-packages.txt`
5. Find `libcrypto3` version, save to: `/opt/course/04/libcrypto-version.txt`

Answer

```
mkdir -p /opt/course/04

# 1. Generate SBOM with bom (SPDX format)
bom generate -o /opt/course/04/sbom.spdx --image nginx:1.25-alpine@sha256:721fa00bc549df26b3e67cc558ff176112d4ba69847537766f3c28e171d180e7

# 2. Generate SBOM with trivy (SPDX-JSON format)
trivy image --format spdx-json --output /opt/course/04/sbom.spdx.json nginx:1.25-alpine@sha256:721fa00bc549df26b3e67cc558ff176112d4ba69847537766f3c28e171d180e7

# 3. Scan SBOM for vulnerabilities
trivy sbom --format json /opt/course/04/sbom.spdx.json > /opt/course/04/sbom-vulns.json

# 4. Query for SSL packages
# ⚠ IMPORTANT: The --fields argument MUST be quoted: --fields 'name,version'
#    Without quotes, bash interprets the comma as a command separator!
bom document query /opt/course/04/sbom.spdx 'name:ssl' --fields 'name,version' > /opt/course/04/ssl-packages.txt
bom document query /opt/course/04/sbom.spdx 'name:openssl' --fields 'name,version' >> /opt/course/04/ssl-packages.txt

# 5. Get libcrypto3 version
bom document query /opt/course/04/sbom.spdx 'name:libcrypto3' --fields 'name,version' > /opt/course/04/libcrypto-version.txt
```

Key Trivy SBOM Commands:

```
# Generate SBOM in different formats
trivy image --format spdx-json --output <path> <image>   # SPDX-JSON
trivy image --format cyclonedx --output <path> <image>   # CycloneDX
trivy image --format spdx --output <path> <image>        # SPDX tag-value

# Scan existing SBOM for vulnerabilities
trivy sbom --format json <sbom-file>                     # JSON output
trivy sbom --format table <sbom-file>                    # Table output
trivy sbom --severity HIGH,CRITICAL <sbom-file>          # Filter by severity
```

Question 5: Create TLS Secret

Weight: 2% | **Domain:** Cluster Setup | **Difficulty:** Very Easy

Question

Create a TLS secret using provided certificate and key files.

Requirements:

- Certificate: `/opt/course/05/tls.crt`
- Key: `/opt/course/05/tls.key`
- Secret name: `my-tls-secret`
- Namespace: `secure-ns`

Save command to: `/opt/course/05/create-secret.txt`

Answer

```
kubectl create secret tls my-tls-secret \
    --cert=/opt/course/05/tls.crt \
    --key=/opt/course/05/tls.key \
    -n secure-ns

echo "kubectl create secret tls my-tls-secret --cert=/opt/course/05/tls.crt --key=/opt/course/05/tls.key -n secure-ns" > /opt/course/05/create-secret.txt
```

Question 6: Docker Daemon Hardening

Weight: 5% | **Domain:** System Hardening | **Difficulty:** Medium

Question

SSH to the cluster node and secure the Docker daemon:

1. Remove user `developer` from the `docker` group
2. Set `/var/run/docker.sock` group ownership to `root`
3. Remove any TCP listener configuration
4. Restart Docker daemon

Save:

- Socket permissions before: `/opt/course/06/socket-before.txt`
- Socket permissions after: `/opt/course/06/socket-after.txt`
- daemon.json content: `/opt/course/06/daemon.json`

Answer

```
ssh <worker-node>

# Save current permissions
ls -la /var/run/docker.sock > /opt/course/06/socket-before.txt

# Remove user from docker group
sudo gpasswd -d developer docker

# Configure Docker daemon
echo '{"group": "root"}' | sudo tee /etc/docker/daemon.json

# Save daemon.json
sudo cp /etc/docker/daemon.json /opt/course/06/daemon.json

# Restart Docker
sudo systemctl restart docker

# Save new permissions
ls -la /var/run/docker.sock > /opt/course/06/socket-after.txt

exit
kubectl get nodes
```

Question 7: Network Policy

Weight: 7% | **Domain:** Cluster Setup | **Difficulty:** Medium-Hard

Question

Create two NetworkPolicies:

1. `deny-all-ingress` in `prod` namespace - deny ALL ingress traffic
2. `allow-from-prod` in `data` namespace - allow ingress ONLY from pods in `prod` namespace with label `env: prod`

Save:

- `/opt/course/07/deny-all-ingress.yaml`
- `/opt/course/07/allow-from-prod.yaml`

Answer

deny-all-ingress.yaml:

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: deny-all-ingress
  namespace: prod
spec:
  podSelector: {}
  policyTypes:
  - Ingress
```

allow-from-prod.yaml:

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-from-prod
  namespace: data
spec:
  podSelector: {}
  policyTypes:
  - Ingress
  ingress:
  - from:
    - namespaceSelector:
        matchLabels:
          env: prod
      podSelector:
        matchLabels:
          env: prod
```

```
kubectl apply -f /opt/course/07/deny-all-ingress.yaml
kubectl apply -f /opt/course/07/allow-from-prod.yaml
```

Question 8: ServiceAccount Token with Projected Volume

Weight: 5% | **Domain:** Cluster Hardening | **Difficulty:** Medium

Question

A security audit has identified a Deployment improperly handling service account tokens.

ServiceAccount `stats-monitor-sa` and Deployment `stats-monitor` exist in namespace `monitoring`.

Task:

1. Disable automatic token mounting on the ServiceAccount
2. Discover the correct audience for the cluster's API Server by inspecting cluster configuration
3. Manually mount the token using a projected volume with:
    - Name: `token`
    - expirationSeconds: `3600`
    - audience: (value discovered in step 2)
    - path: `token`
    - Mount at: `/var/run/secrets/kubernetes.io/serviceaccount/token`
    - Mount must be read-only

Note: The Deployment manifest can be found at `~/stats-monitor/deployment.yaml`

Answer

Step 1: Discover the API Server audience

```
# Method 1: Check OIDC configuration (most reliable)
kubectl get --raw /.well-known/openid-configuration | jq -r '.issuer'

# Method 2: Check API server flags
kubectl -n kube-system get pod kube-apiserver-<node> -o yaml | grep service-account-issuer

# Typical output: https://kubernetes.default.svc.cluster.local
```

Step 2: Modify ServiceAccount

```
# Option 1: Patch (quickest)
kubectl patch sa stats-monitor-sa -n monitoring -p '{"automountServiceAccountToken": false}'

# Option 2: Edit directly
kubectl edit sa stats-monitor-sa -n monitoring
# Add: automountServiceAccountToken: false
```

serviceaccount.yaml:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: stats-monitor-sa
  namespace: monitoring
automountServiceAccountToken: false
```

Step 3: Modify Deployment

```
# Edit the deployment manifest
vi ~/stats-monitor/deployment.yaml

# Apply changes
kubectl apply -f ~/stats-monitor/deployment.yaml
```

deployment.yaml:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: stats-monitor
  namespace: monitoring
spec:
  replicas: 1
  selector:
    matchLabels:
      app: stats-monitor
  template:
    metadata:
      labels:
        app: stats-monitor
    spec:
      serviceAccountName: stats-monitor-sa
      containers:
      - name: stats
        image: busybox:1.36
        command: ["sleep", "3600"]
        volumeMounts:
        - name: token
          mountPath: /var/run/secrets/kubernetes.io/serviceaccount
          readOnly: true
      volumes:
      - name: token
        projected:
          sources:
          - serviceAccountToken:
              expirationSeconds: 3600
              path: token
              audience: https://kubernetes.default.svc.cluster.local
```

Step 4: Verify

```
# Verify ServiceAccount
kubectl get sa stats-monitor-sa -n monitoring -o yaml | grep
automountServiceAccountToken

# Verify projected volume configuration
kubectl get deployment stats-monitor -n monitoring -o yaml | grep -A8
"serviceAccountToken"

# Verify token is mounted in pod
kubectl exec -n monitoring deployment/stats-monitor -- ls -la
/var/run/secrets/kubernetes.io/serviceaccount/

# Check token content
kubectl exec -n monitoring deployment/stats-monitor -- cat
/var/run/secrets/kubernetes.io/serviceaccount/token | head -c 50
```

Key Points:

- **Audience discovery**: Not provided in exam - must discover via cluster inspection
- **Audience field**: Validates token is intended for specific cluster API server
- **expirationSeconds**: Token auto-rotates before expiration
- **readOnly: true**: Prevents token modification

⚠ CRITICAL - How path and mountPath Work Together:

- **mountPath**: The directory where the volume is mounted (`/var/run/secrets/kubernetes.io/serviceaccount`)
- **path**: The filename created **INSIDE** that mounted directory (`token`)
- **Final location**: `mountPath + path = /var/run/secrets/kubernetes.io/serviceaccount/token`
- **DO NOT use subPath**: Not in official docs, prevents auto-rotation, not needed with projected volumes
- Projected volumes can combine multiple sources (SA token + configMap + secret) into one directory

---

Question 9: Configure Kubernetes Auditing

Weight: 7% | **Domain:** Monitoring, Logging and Runtime Security | **Difficulty:** Medium-Hard

Question

Configure the kube-apiserver to enable auditing:

1. Create audit policy at `/etc/kubernetes/audit/policy.yaml`:
    - Log `secrets` at `Metadata` level
    - Log `configmaps` at `Metadata` level
    - Log `namespaces` at `RequestResponse` level

2. Configure API server with:
   - `--audit-policy-file=/etc/kubernetes/audit/policy.yaml`
   - `--audit-log-path=/var/log/kubernetes/audit/audit.log`
   - `--audit-log-maxage=2`
   - `--audit-log-maxbackup=10`

Save: `/opt/course/09/audit-policy.yaml`

Answer

audit-policy.yaml:

```
apiVersion: audit.k8s.io/v1
kind: Policy
rules:
  - level: Metadata
    resources:
    - group: ""
      resources: ["secrets"]

  - level: Metadata
    resources:
    - group: ""
      resources: ["configmaps"]

  - level: RequestResponse
    resources:
    - group: ""
      resources: ["namespaces"]

  - level: Metadata
    omitStages:
    - RequestReceived
```

```
sudo mkdir -p /etc/kubernetes/audit
sudo mkdir -p /var/log/kubernetes/audit

# Save audit policy
cat << 'POLICY' | sudo tee /etc/kubernetes/audit/policy.yaml
apiVersion: audit.k8s.io/v1
kind: Policy
rules:
  - level: Metadata
    resources:
    - group: ""
      resources: ["secrets"]
  - level: Metadata
    resources:
    - group: ""
      resources: ["configmaps"]
  - level: RequestResponse
    resources:
    - group: ""
      resources: ["namespaces"]
  - level: Metadata
    omitStages:
    - RequestReceived
POLICY

# Edit kube-apiserver manifest and add:
# Flags:
#   - --audit-policy-file=/etc/kubernetes/audit/policy.yaml
#   - --audit-log-path=/var/log/kubernetes/audit/audit.log
#   - --audit-log-maxage=2
#   - --audit-log-maxbackup=10
#
# volumeMounts:
#   - mountPath: /etc/kubernetes/audit
#     name: audit-policy
#     readOnly: true
#   - mountPath: /var/log/kubernetes/audit
#     name: audit-log
#     # readOnly: false (or omit - API server must WRITE audit logs)
#
# volumes:
#   - hostPath:
#       path: /etc/kubernetes/audit
#       type: DirectoryOrCreate
#     name: audit-policy
#   - hostPath:
#       path: /var/log/kubernetes/audit
#       type: DirectoryOrCreate
#     name: audit-log
```

⚠ CRITICAL: Common Mistakes to Avoid

❌ **WRONG - Mounting individual files:**

```
volumeM ounts:
  - mountPath: /etc/kubernetes/audit/policy.yaml  # ❌ File, not
directory!
    name: audit-policy
  - mountPath: /var/log/kubernetes/audit/audit.log  # ❌ File, not
directory!
    name: audit-log
    readOnly: true  # ❌ API server can't write!
```

✅ **CORRECT - Mount directories:**

```
volumeMounts:
  - mountPath: /etc/kubernetes/audit  # ✅ Directory
    name: audit-policy
    readOnly: true  # ✅ Policy is read-only
  - mountPath: /var/log/kubernetes/audit  # ✅ Directory
    name: audit-log
    # readOnly: false or omit - API server writes logs here
```

Key Points:

- Always mount **directories**, never individual files for auditing
- Audit policy directory must be `readOnly: true`
- Audit log directory must be writable (omit readOnly or set to false)
- Error "is a directory" = you tried to mount a file path
- API server will fail to start if audit log path is not writable

---

Question 10: ImagePolicyWebhook Admission Controller

Weight: 7% | **Domain:** Supply Chain Security | **Difficulty:** Medium-Hard

Question

Fix and complete an existing ImagePolicyWebhook configuration at `/etc/kubernetes/epconfig/`:

1. Fix `defaultAllow` from `true` to `false` (fail-closed)
2. Fix the webhook server URL (currently a placeholder)
3. Fix `current-context` (currently empty)
4. Enable `ImagePolicyWebhook` in `--enable-admission-plugins`
5. Set `--admission-control-config-file`
6. Test the webhook by trying to create a pod (should be DENIED)

Save:

- `/opt/course/10/admission_config.yaml`
- `/opt/course/10/kubeconfig.yaml`
- `/opt/course/10/webhook-test.txt` (error message from test)

Answer

Fix admission_config.yaml:

```
apiVersion: apiserver.config.k8s.io/v1
kind: AdmissionConfiguration
plugins:
- name: ImagePolicyWebhook
  configuration:
    imagePolicy:
      kubeConfigFile: /etc/kubernetes/epconfig/kubeconfig.yaml
      allowTTL: 50
      denyTTL: 50
      retryBackoff: 500
      defaultAllow: false  # Changed from true to false!
```

Fix kubeconfig.yaml:

```
apiVersion: v1
kind: Config
clusters:
- name: image-policy-webhook
  cluster:
    server: https://image-policy-webhook.default.svc:443/image_policy  # Fixed URL
    insecure-skip-tls-verify: true
users:
- name: api-server
  user: {}
contexts:
- name: default
  context:
    cluster: image-policy-webhook
    user: api-server
current-context: default  # ⚠ CRITICAL: Fixed from empty ""!
```

> ⚠ **EXAM ALERT:** The `current-context` field is **MANDATORY** and often forgotten! If `current-context` is empty (`""`) or missing, the ImagePolicyWebhook will **NOT work** and the API server will fail. Pay special attention to this field in the exam!

API server manifest changes:

```
# Add to --enable-admission-plugins:
- --enable-admission-plugins=NodeRestriction,ImagePolicyWebhook

# Add flag:
- --admission-control-config-file=/etc/kubernetes/epconfig/admission_config.yaml

# Add volumeMount:
- mountPath: /etc/kubernetes/epconfig
  name: epconfig
  readOnly: true

# Add volume:
- hostPath:
    path: /etc/kubernetes/epconfig
    type: DirectoryOrCreate
  name: epconfig
```

Test the ImagePolicyWebhook:

```
# Try to create a test pod - should be DENIED!
kubectl run test-pod --image=nginx 2>&1 | tee /opt/course/10/webhook-test.txt

# Expected: Error (Forbidden) - because defaultAllow=false and webhook unreachable
# This PROVES fail-closed behavior is working correctly!
```

Question 11: Pod Security Admission

Weight: 7% | **Domain:** Minimize Microservice Vulnerabilities | **Difficulty:** Medium

Question

Namespace `team-blue` has PSA configured with `restricted` level. Some pods don't comply.

Task:

1. Use `kubectl label --dry-run=server` to identify violations
2. Delete non-compliant pods
3. Keep compliant pods running

Save:

- Warning output: `/opt/course/11/violations.txt`
- Deleted pod names: `/opt/course/11/deleted-pods.txt`

- Command used: `/opt/course/11/command.txt`

Answer

```
# Identify violations
kubectl label --dry-run=server --overwrite ns team-blue \
    pod-security.kubernetes.io/enforce=restricted 2>&1 | tee
/opt/course/11/violations.txt

# Save command
echo 'kubectl label --dry-run=server --overwrite ns team-blue pod-
security.kubernetes.io/enforce=restricted' > /opt/course/11/command.txt

# Delete non-compliant pods (based on warning output)
kubectl delete pod hostnetwork-pod -n team-blue
kubectl delete pod root-pod -n team-blue
kubectl delete pod escalation-pod -n team-blue

# Save deleted pods
cat << 'EOF' > /opt/course/11/deleted-pods.txt
hostnetwork-pod
root-pod
escalation-pod
EOF
```

Question 12: Dockerfile and Deployment Security

Weight: 7% | **Domain:** Supply Chain Security | **Difficulty:** Medium

Question

Fix security issues in Dockerfile and Deployment manifest.

Dockerfile issues:

- Running as root
- Using `latest` tag
- Using `ADD` instead of `COPY`

Deployment issues:

- privileged: true
- allowPrivilegeEscalation: true
- No runAsNonRoot
- No readOnlyRootFilesystem

Save:

- `/opt/course/12/Dockerfile-fixed`
- `/opt/course/12/deployment-fixed.yaml`

Answer

Dockerfile-fixed:

```dockerfile
FROM nginx:1.25.3-alpine

COPY config.txt /etc/config.txt
COPY index.html /usr/share/nginx/html/

ADD app.tar.gz /app

RUN addgroup -g 1001 appgroup && \
    adduser -u 1001 -G appgroup -D appuser && \
    chown -R appuser:appgroup /usr/share/nginx/html /var/cache/nginx /var/run

USER appuser

EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]
```

deployment-fixed.yaml:

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web-app
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: web-app
  template:
    metadata:
      labels:
        app: web-app
    spec:
      securityContext:
        runAsNonRoot: true
        runAsUser: 1001
        fsGroup: 1001
      containers:
      - name: nginx
        image: nginx:1.25.3-alpine
        ports:
        - containerPort: 80
        securityContext:
          privileged: false
          allowPrivilegeEscalation: false
          readOnlyRootFilesystem: true
          capabilities:
            drop:
            - ALL
        volumeMounts:
        - name: tmp
          mountPath: /tmp
        - name: cache
          mountPath: /var/cache/nginx
        - name: run
          mountPath: /var/run
      volumes:
      - name: tmp
        emptyDir: {}
      - name: cache
        emptyDir: {}
      - name: run
        emptyDir: {}
```

Question 13: Kubelet Security Configuration

Weight: 5% | **Domain:** Cluster Hardening | **Difficulty:** Medium

Question

SSH to the worker node and secure `/var/lib/kubelet/config.yaml`:

1. Set `authentication.anonymous.enabled` to `false`
2. Set `authentication.webhook.enabled` to `true`
3. Set `authorization.mode` to `Webhook`
4. Restart kubelet

Save:

- Before: `/opt/course/13/kubelet-before.yaml`
- After: `/opt/course/13/kubelet-after.yaml`

Answer

```
ssh <worker-node>

mkdir -p /opt/course/13
sudo cp /var/lib/kubelet/config.yaml /opt/course/13/kubelet-before.yaml

# Edit kubelet config
sudo vi /var/lib/kubelet/config.yaml
```

Kubelet config changes:

```
authentication:
  anonymous:
    enabled: false
  webhook:
    cacheTTL: 0s
    enabled: true
  x509:
    clientCAFile: /etc/kubernetes/pki/ca.crt
authorization:
  mode: Webhook
  webhook:
    cacheAuthorizedTTL: 0s
    cacheUnauthorizedTTL: 0s
```

```
# Restart kubelet
sudo systemctl daemon-reload
sudo systemctl restart kubelet

# Save config
sudo cp /var/lib/kubelet/config.yaml /opt/course/13/kubelet-after.yaml

exit
kubectl get nodes
```

Question 14: Ensure Container Immutability

Weight: 7% | **Domain:** Monitoring, Logging and Runtime Security | **Difficulty:** Medium

Question

Modify Deployment `nginx` in namespace `immutable-ns` to make the container filesystem immutable:

1. Add `readOnlyRootFilesystem: true`
2. Add `emptyDir` volumes for writable paths nginx requires

Save: `/opt/course/14/deployment-immutable.yaml`

Answer

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
  namespace: immutable-ns
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.25-alpine
        ports:
        - containerPort: 80
        securityContext:
          readOnlyRootFilesystem: true
        volumeMounts:
        - name: cache
          mountPath: /var/cache/nginx
        - name: run
          mountPath: /var/run
        - name: tmp
          mountPath: /tmp
      volumes:
      - name: cache
        emptyDir: {}
      - name: run
        emptyDir: {}
      - name: tmp
        emptyDir: {}
```

```
kubectl apply -f /opt/course/14/deployment-immutable.yaml
```

Question 15: Containerd Security Hardening

Weight: 5% | **Domain:** System Hardening | **Difficulty:** Medium

Question

SSH to the worker node and secure containerd:

1. Remove user `developer` from container-related groups
2. Set `/run/containerd/containerd.sock` group ownership to `root`
3. Remove any TCP listener configuration
4. Restart containerd

Save:

- Socket permissions before: `/opt/course/15/socket-before.txt`
- Socket permissions after: `/opt/course/15/socket-after.txt`
- config.toml content: `/opt/course/15/config.toml`
- TCP port proof: `/opt/course/15/netstat-after.txt`

Answer

```
ssh <worker-node>

mkdir -p /opt/course/15

# Save current permissions
ls -la /run/containerd/containerd.sock > /opt/course/15/socket-before.txt

# Remove user from containerd group
sudo gpasswd -d developer containerd

# Fix socket ownership
sudo chown root:root /run/containerd/containerd.sock
sudo chmod 660 /run/containerd/containerd.sock

# Remove TCP listener
sudo sed -i '/tcp_address/d' /etc/containerd/config.toml

# Save config
sudo cp /etc/containerd/config.toml /opt/course/15/config.toml

# Restart containerd
sudo systemctl restart containerd

# Save results
ls -la /run/containerd/containerd.sock > /opt/course/15/socket-after.txt
ss -tlnp > /opt/course/15/netstat-after.txt

exit
kubectl get nodes
```