CKA Practice Questions - Complete Solutions

Table of Contents

1. [Question 1: Kubeconfig Extraction](#question-1-kubeconfig-extraction)
2. [Question 2: MinIO Operator Installation](#question-2-minio-operator-installation)
3. [Question 3: Scale Down Pods](#question-3-scale-down-pods)
4. [Question 4: Pod Termination Priority](#question-4-pod-termination-priority)
5. [Question 5: HorizontalPodAutoscaler (HPA)](#question-5-horizontalpodautoscaler-hpa)
6. [Question 6: PersistentVolume and Deployment](#question-6-persistentvolume-and-deployment)
7. [Question 7: Metrics Server Scripts](#question-7-metrics-server-scripts)
8. [Question 8: Node Upgrade and Join Cluster](#question-8-node-upgrade-and-join-cluster)
9. [Question 9: ServiceAccount API Query](#question-9-serviceaccount-api-query)
10. [Question 10: RBAC - ServiceAccount, Role, RoleBinding](#question-10-rbac-serviceaccount-role-rolebinding)
11. [Question 11: DaemonSet Creation](#question-11-daemonset-creation)
12. [Question 12: Deployment with Pod Anti-Affinity](#question-12-deployment-with-pod-anti-affinity)
13. [Question 13: Gateway API HTTPRoute](#question-13-gateway-api-httproute)
14. [Question 14: Certificate Management](#question-14-certificate-management)
15. [Question 15: NetworkPolicy](#question-15-networkpolicy)
16. [Question 16: CoreDNS Configuration](#question-16-coredns-configuration)
17. [Question 17: Container Runtime (crictl)](#question-17-container-runtime-crictl)

Question 1: Kubeconfig Extraction

Server: `ssh cka9412`

Task:
Extract the following information from kubeconfig file `/opt/course/1/kubeconfig`:
• Write all kubeconfig context names into `/opt/course/1/contexts`, one per line
• Write the name of the current context into `/opt/course/1/current-context`
• Write the client-certificate of user account-0027 base64-decoded into `/opt/course/1/cert`

Solution

```
ssh cka9412

# Part 1: Extract all context names (one per line)
kubectl config get-contexts -o name --kubeconfig=/opt/course/1/kubeconfig > /opt/course/1/contexts

# Part 2: Get the current context
kubectl config current-context --kubeconfig=/opt/course/1/kubeconfig > /opt/course/1/current-context

# Part 3: Extract and decode the client-certificate for user account-0027
kubectl config view --kubeconfig=/opt/course/1/kubeconfig --raw -o jsonpath='{.users[?(@.name=="account-0027")].use

# Verification
cat /opt/course/1/contexts
cat /opt/course/1/current-context
head /opt/course/1/cert
```

Key Points:
• Use `--raw` to see base64-encoded data
• Use jsonpath to filter for specific user
• Pipe through `base64 -d` to decode

Question 2: MinIO Operator Installation

Server: `ssh cka7968`

Task:
Install the MinIO Operator using Helm in Namespace `minio`:
1. Create Namespace minio
2. Install Helm chart `minio/operator` with release name `minio-operator`
3. Update Tenant resource to include `enableSFTP: true` under features
4. Create the Tenant resource from `/opt/course/2/minio-tenant.yaml`

Solution

```
ssh cka7968

# Step 1: Create the minio namespace
kubectl create namespace minio

# Step 2: Add the MinIO Helm repository
helm repo add minio https://operator.min.io/
helm repo update

# Step 3: Install the MinIO Operator Helm chart
helm install minio-operator minio/operator --namespace minio

# Step 4: Wait for operator to be ready
kubectl wait --for=condition=ready pod -l app.kubernetes.io/name=operator -n minio --timeout=300s

# Step 5: Update the Tenant YAML file
sed -i '/features:/a\    enableSFTP: true' /opt/course/2/minio-tenant.yaml

# Step 6: Apply the Tenant resource
kubectl apply -f /opt/course/2/minio-tenant.yaml

# Verification
helm list -n minio
kubectl get tenants -n minio
```

Key Points:
• Helm release name must be `minio-operator`
• Use `sed` to add `enableSFTP: true` under features section
• Verify installation with `helm list` and `kubectl get tenants`

Question 3: Scale Down Pods

Server: `ssh cka3962`

Task:
Scale down the two `o3db-*` Pods in Namespace `project-h800` to one replica.

Solution

```
ssh cka3962

# Check pods
kubectl get pods -n project-h800 | grep o3db

# Identify the controller type
kubectl get deployments,statefulsets -n project-h800 | grep o3db

# Scale down (if it's a deployment)
kubectl scale deployment o3db -n project-h800 --replicas=1

# Or find by pattern and scale all
kubectl get deploy,sts -n project-h800 -o name | grep o3db | xargs -I {} kubectl scale {} -n project-h800 --replica

# Verification
kubectl get pods -n project-h800 | grep o3db
```

Key Points:
• Identify the controller (Deployment or StatefulSet)
• Use `kubectl scale` to adjust replicas
• Verify only 1 pod remains running

Question 4: Pod Termination Priority

Server: `ssh cka2556`

Task:
Check all Pods in Namespace `project-c13` and find names of those that would be terminated first if nodes run out of resources. Write Pod names into `/opt/course/4/pods-terminated-first.txt`.

Solution

```
ssh cka2556

# Get pods with QoS and Priority information
kubectl get pods -n project-c13 -o json | jq -r '.items | sort_by(.spec.priority // 0,
    (if .status.qosClass == "BestEffort" then 0
     elif .status.qosClass == "Burstable" then 1
     else 2 end)) |
    .[].metadata.name' | head -n 2 > /opt/course/4/pods-terminated-first.txt

# Verify
cat /opt/course/4/pods-terminated-first.txt
```

Pod Eviction Order:
1. **BestEffort** QoS (no requests/limits) - terminated FIRST
2. **Burstable** QoS (partial requests/limits) - terminated SECOND
3. **Guaranteed** QoS (requests = limits) - terminated LAST
4. Within same QoS, lower priority terminated first

Question 5: HorizontalPodAutoscaler (HPA)

Server: `ssh cka5774`

Task:
Using Kustomize config at `/opt/course/5/api-gateway`:
• Remove ConfigMap `horizontal-scaling-config`
• Add HPA named `api-gateway` with min 2, max 4 replicas, 50% CPU
• In prod, HPA should have max 6 replicas
• Apply changes for staging and prod

Solution

```
ssh cka5774

cd /opt/course/5/api-gateway

# Remove ConfigMap reference from base/kustomization.yaml
cd base
sed -i '/horizontal-scaling-config/d' kustomization.yaml

# Create HPA in base
cat > hpa.yaml <<'EOF'
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: api-gateway
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: api-gateway
  minReplicas: 2
  maxReplicas: 4
  metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 50
EOF

# Add hpa.yaml to resources
echo "  - hpa.yaml" >> kustomization.yaml

# Create prod patch
cd ../prod
cat > hpa-patch.yaml <<'EOF'
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: api-gateway
spec:
  maxReplicas: 6
EOF

# Add patch to prod kustomization
echo "patchesStrategicMerge:" >> kustomization.yaml
echo "  - hpa-patch.yaml" >> kustomization.yaml

# Apply changes
cd /opt/course/5/api-gateway
kubectl kustomize staging/ | kubectl apply -f -
kubectl kustomize prod/ | kubectl apply -f -

# Verification
kubectl get hpa api-gateway -n api-gateway-staging
```

```
kubectl get hpa api-gateway -n api-gateway-prod
```

Question 6: PersistentVolume and Deployment

Server: `ssh cka7968`

Task:
Create:
1. PersistentVolume `safari-pv`: 2Gi, ReadWriteOnce, hostPath `/Volumes/Data`
2. PersistentVolumeClaim `safari-pvc` in Namespace `project-t230`: 2Gi, ReadWriteOnce
3. Deployment `safari` mounting volume at `/tmp/safari-data`, image `httpd:2-alpine`

Solution

```
ssh cka7968

# Create PersistentVolume
kubectl apply -f - <<EOF
apiVersion: v1
kind: PersistentVolume
metadata:
  name: safari-pv
spec:
  capacity:
    storage: 2Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: /Volumes/Data
EOF

# Ensure namespace exists
kubectl create namespace project-t230 --dry-run=client -o yaml | kubectl apply -f -

# Create PersistentVolumeClaim
kubectl apply -f - <<EOF
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: safari-pvc
  namespace: project-t230
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 2Gi
EOF

# Create Deployment
kubectl apply -f - <<EOF
apiVersion: apps/v1
kind: Deployment
metadata:
  name: safari
  namespace: project-t230
spec:
  replicas: 1
  selector:
    matchLabels:
      app: safari
  template:
    metadata:
      labels:
        app: safari
    spec:
      containers:
      - name: safari
        image: httpd:2-alpine
        volumeMounts:
```

```
        - name: safari-volume
          mountPath: /tmp/safari-data
      volumes:
      - name: safari-volume
        persistentVolumeClaim:
          claimName: safari-pvc
EOF

# Verification
kubectl get pv safari-pv
kubectl get pvc safari-pvc -n project-t230
kubectl get deployment safari -n project-t230
```

Question 7: Metrics Server Scripts

Server: `ssh cka5774`

Task:
Create two bash scripts:
• `/opt/course/7/node.sh` - show resource usage of nodes
• `/opt/course/7/pod.sh` - show resource usage of Pods and containers

Solution

```
ssh cka5774

# Create directory
mkdir -p /opt/course/7

# Create node.sh
cat > /opt/course/7/node.sh <<'EOF'
#!/bin/bash
kubectl top nodes
EOF

# Create pod.sh
cat > /opt/course/7/pod.sh <<'EOF'
#!/bin/bash
kubectl top pods --all-namespaces --containers
EOF

# Make scripts executable
chmod +x /opt/course/7/node.sh
chmod +x /opt/course/7/pod.sh

# Test scripts
/opt/course/7/node.sh
/opt/course/7/pod.sh
```

Question 8: Node Upgrade and Join Cluster

Server: `ssh cka3962`

Task:
• Update `cka3962-node1` Kubernetes to exact version of control plane
• Add the node to cluster using kubeadm

Solution

```
ssh cka3962

# Get control plane version
KUBE_VERSION=$(kubectl version -o json | jq -r '.serverVersion.gitVersion' | sed 's/v//')

# Generate join command
JOIN_CMD=$(sudo kubeadm token create --print-join-command)

# SSH to worker node
ssh cka3962-node1

# Update apt cache
sudo apt-get update

# Find matching package version
apt-cache madison kubeadm | grep ${KUBE_VERSION%-*}

# Set package version (adjust based on available packages)
PACKAGE_VERSION="1.29.0-1.1"  # Example

# Upgrade kubeadm
sudo apt-mark unhold kubeadm
sudo apt-get install -y kubeadm=${PACKAGE_VERSION}
sudo apt-mark hold kubeadm

# Upgrade kubelet and kubectl
sudo apt-mark unhold kubelet kubectl
sudo apt-get install -y kubelet=${PACKAGE_VERSION} kubectl=${PACKAGE_VERSION}
sudo apt-mark hold kubelet kubectl

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

# Join cluster (use actual join command from control plane)
sudo kubeadm join <ip>:6443 --token <token> --discovery-token-ca-cert-hash sha256:<hash>

exit

# Verify on control plane
kubectl get nodes
kubectl wait --for=condition=Ready node/cka3962-node1 --timeout=300s
```

Question 9: ServiceAccount API Query

Server: `ssh cka9412`

Task:
• Create Pod `api-contact` using ServiceAccount `secret-reader` in Namespace `project-swan`
• Exec into Pod and use curl to query all Secrets from Kubernetes API
• Write result to `/opt/course/9/result.json`

Solution

```
ssh cka9412

# Create Pod
kubectl apply -f - <<EOF
apiVersion: v1
kind: Pod
metadata:
  name: api-contact
  namespace: project-swan
spec:
  serviceAccountName: secret-reader
  containers:
  - name: nginx
    image: nginx:1-alpine
    command: ["/bin/sh", "-c", "sleep 3600"]
EOF

# Wait for Pod
kubectl wait --for=condition=ready pod/api-contact -n project-swan --timeout=120s

# Query API and save result
kubectl exec -n project-swan api-contact -- sh -c '
  apk add --no-cache curl 2>/dev/null || true
  TOKEN=$(cat /var/run/secrets/kubernetes.io/serviceaccount/token)
  CACERT=/var/run/secrets/kubernetes.io/serviceaccount/ca.crt
  curl -s --cacert ${CACERT} \
       -H "Authorization: Bearer ${TOKEN}" \
       https://kubernetes.default.svc/api/v1/secrets
' > /opt/course/9/result.json

# Verify
cat /opt/course/9/result.json | head -20
```

Question 10: RBAC - ServiceAccount, Role, RoleBinding

Server: `ssh cka3962`

Task:
Create in Namespace `project-hamster`:
• ServiceAccount `processor`
• Role `processor` allowing only create Secrets and ConfigMaps
• RoleBinding `processor`

Solution

```
ssh cka3962

# Create ServiceAccount
kubectl create serviceaccount processor -n project-hamster

# Create Role
kubectl create role processor \
  --verb=create \
  --resource=secrets,configmaps \
  -n project-hamster

# Create RoleBinding
kubectl create rolebinding processor \
  --role=processor \
  --serviceaccount=project-hamster:processor \
  -n project-hamster

# Verification
kubectl describe sa processor -n project-hamster
kubectl describe role processor -n project-hamster
kubectl describe rolebinding processor -n project-hamster

# Test permissions
kubectl auth can-i create secrets --as=system:serviceaccount:project-hamster:processor -n project-hamster
kubectl auth can-i create configmaps --as=system:serviceaccount:project-hamster:processor -n project-hamster
```

Question 11: DaemonSet Creation

Server: `ssh cka2556`

Task:
Create DaemonSet `ds-important` in Namespace `project-tiger`:
• Image: `httpd:2-alpine`
• Labels: `id=ds-important`, `uuid=18426a0b-5f59-4e10-923f-c0e078e82462`
• Resources: 10m CPU, 10Mi memory
• Run on all nodes including control planes

Solution

```
ssh cka2556

kubectl apply -f - <<EOF
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: ds-important
  namespace: project-tiger
  labels:
    id: ds-important
    uuid: 18426a0b-5f59-4e10-923f-c0e078e82462
spec:
  selector:
    matchLabels:
      id: ds-important
      uuid: 18426a0b-5f59-4e10-923f-c0e078e82462
  template:
    metadata:
      labels:
        id: ds-important
        uuid: 18426a0b-5f59-4e10-923f-c0e078e82462
    spec:
      tolerations:
      - key: node-role.kubernetes.io/control-plane
        operator: Exists
        effect: NoSchedule
      - key: node-role.kubernetes.io/master
        operator: Exists
        effect: NoSchedule
      containers:
      - name: httpd
        image: httpd:2-alpine
        resources:
          requests:
            cpu: 10m
            memory: 10Mi
EOF

# Verification
kubectl get ds ds-important -n project-tiger
kubectl get pods -n project-tiger -l id=ds-important -o wide
```

Question 12: Deployment with Pod Anti-Affinity

Server: `ssh cka2556`

Task:
Create Deployment `deploy-important` in Namespace `project-tiger`:
• 3 replicas
• Label: `id=very-important`
• Container1: `nginx:1-alpine`, Container2: `google/pause`
• Only one pod per node (topologyKey: kubernetes.io/hostname)

Solution

```
ssh cka2556

kubectl apply -f - <<EOF
apiVersion: apps/v1
kind: Deployment
metadata:
  name: deploy-important
  namespace: project-tiger
  labels:
    id: very-important
spec:
  replicas: 3
  selector:
    matchLabels:
      id: very-important
  template:
    metadata:
      labels:
        id: very-important
    spec:
      affinity:
        podAntiAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
          - labelSelector:
              matchLabels:
                id: very-important
            topologyKey: kubernetes.io/hostname
      containers:
      - name: container1
        image: nginx:1-alpine
      - name: container2
        image: google/pause
EOF

# Verification
kubectl get deployment deploy-important -n project-tiger
kubectl get pods -n project-tiger -l id=very-important -o wide
```

Expected Result: 2 pods Running, 1 pod Pending (only 2 worker nodes)

Question 13: Gateway API HTTPRoute

Server: `ssh cka7968`

Task:
In Namespace `project-r500`:
• Create HTTPRoute `traffic-director` replicating routes from old Ingress
• Add path `/auto` that redirects based on User-Agent header

Solution

```
ssh cka7968

# Check old Ingress
cat /opt/course/13/ingress.yaml

# Get Gateway name
GATEWAY_NAME=$(kubectl get gateway -n project-r500 -o jsonpath='{.items[0].metadata.name}')

# Create HTTPRoute
kubectl apply -f - <<EOF
apiVersion: gateway.networking.k8s.io/v1
kind: HTTPRoute
metadata:
  name: traffic-director
  namespace: project-r500
spec:
  parentRefs:
  - name: ${GATEWAY_NAME}
  hostnames:
  - "r500.gateway"
  rules:
  - matches:
    - path:
        type: PathPrefix
        value: /desktop
    backendRefs:
    - name: desktop
      port: 80
  - matches:
    - path:
        type: PathPrefix
        value: /mobile
    backendRefs:
    - name: mobile
      port: 80
  - matches:
    - path:
        type: PathPrefix
        value: /auto
      headers:
      - name: User-Agent
        value: mobile
        type: Exact
    backendRefs:
    - name: mobile
      port: 80
  - matches:
    - path:
        type: PathPrefix
        value: /auto
    backendRefs:
    - name: desktop
      port: 80
EOF

# Test
curl http://r500.gateway:30080/desktop
curl http://r500.gateway:30080/mobile
```

```
curl http://r500.gateway:30080/auto -H "User-Agent: mobile"
curl http://r500.gateway:30080/auto
```

Question 14: Certificate Management

Server: `ssh cka9412`

Task:
• Check kube-apiserver certificate expiration with openssl
• Write expiration date to `/opt/course/14/expiration`
• Verify with kubeadm
• Create renewal script at `/opt/course/14/kubeadm-renew-certs.sh`

Solution

```
ssh cka9412

# Create directory
sudo mkdir -p /opt/course/14

# Check and save expiration with openssl
sudo openssl x509 -in /etc/kubernetes/pki/apiserver.crt -noout -enddate | cut -d= -f2 | sudo tee /opt/course/14/exp

# Verify with kubeadm
sudo kubeadm certs check-expiration

# Create renewal script
sudo tee /opt/course/14/kubeadm-renew-certs.sh > /dev/null <<'EOF'
#!/bin/bash
kubeadm certs renew apiserver
EOF

sudo chmod +x /opt/course/14/kubeadm-renew-certs.sh

# Verification
cat /opt/course/14/expiration
cat /opt/course/14/kubeadm-renew-certs.sh
```

Question 15: NetworkPolicy

Server: `ssh cka7968`

Task:
Create NetworkPolicy `np-backend` in Namespace `project-snake`:
• Apply to `backend-*` Pods
• Allow connection to `db1-*` on port 1111
• Allow connection to `db2-*` on port 2222
• Block all other connections (e.g., vault:3333)

Solution

```
ssh cka7968

kubectl apply -f - <<EOF
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: np-backend
  namespace: project-snake
spec:
  podSelector:
    matchLabels:
      app: backend
  policyTypes:
  - Egress
  egress:
  - to:
    - podSelector:
        matchLabels:
          app: db1
    ports:
    - protocol: TCP
      port: 1111
  - to:
    - podSelector:
        matchLabels:
          app: db2
    ports:
    - protocol: TCP
      port: 2222
EOF

# Verification
kubectl describe networkpolicy np-backend -n project-snake

# Test connectivity
BACKEND_POD=$(kubectl get pods -n project-snake -l app=backend -o jsonpath='{.items[0].metadata.name}')
DB1_IP=$(kubectl get pods -n project-snake -l app=db1 -o jsonpath='{.items[0].status.podIP}')
kubectl -n project-snake exec $BACKEND_POD -- curl -m 2 $DB1_IP:1111
```

Question 16: CoreDNS Configuration

Server: `ssh cka5774`

Task:
• Backup CoreDNS config to `/opt/course/16/coredns_backup.yaml`
• Update CoreDNS to support `custom-domain` in addition to `cluster.local`
• Test DNS resolution works for both domains

Solution

```
ssh cka5774

# Backup
mkdir -p /opt/course/16
kubectl get configmap coredns -n kube-system -o yaml > /opt/course/16/coredns_backup.yaml

# Update configuration
kubectl get configmap coredns -n kube-system -o yaml > /tmp/coredns-update.yaml

# Add custom-domain to kubernetes line
sed -i '/kubernetes cluster\.local/s/cluster\.local/cluster.local custom-domain/' /tmp/coredns-update.yaml

# Apply
kubectl apply -f /tmp/coredns-update.yaml

# Restart CoreDNS
kubectl delete pod -n kube-system -l k8s-app=kube-dns

# Wait for ready
kubectl wait --for=condition=ready pod -n kube-system -l k8s-app=kube-dns --timeout=60s

# Test
kubectl run test1 --image=busybox:1 --rm -i --restart=Never -- nslookup kubernetes.default.svc.cluster.local
kubectl run test2 --image=busybox:1 --rm -i --restart=Never -- nslookup kubernetes.default.svc.custom-domain
```

Configuration Change:

```
# From:
kubernetes cluster.local in-addr.arpa ip6.arpa {

# To:
kubernetes cluster.local custom-domain in-addr.arpa ip6.arpa {
```

Question 17: Container Runtime (crictl)

Server: `ssh cka2556`

Task:
• Create Pod `tigers-reunite` with image `httpd:2-alpine` and labels in Namespace `project-tiger`
• SSH to node and find container with crictl
• Write container ID and runtimeType to `/opt/course/17/pod-container.txt`
• Write logs to `/opt/course/17/pod-container.log`

Solution

```
ssh cka2556

# Create Pod
kubectl run tigers-reunite \
  --image=httpd:2-alpine \
  --namespace=project-tiger \
  --labels=pod=container,container=pod

# Wait for Pod
kubectl wait --for=condition=ready pod/tigers-reunite -n project-tiger --timeout=60s

# Get node name
NODE=$(kubectl get pod tigers-reunite -n project-tiger -o jsonpath='{.spec.nodeName}')

# SSH to node
ssh $NODE << 'ENDSSH'

# Find container
CONTAINER_ID=$(sudo crictl ps | grep tigers-reunite | grep httpd | awk '{print $1}')
echo "Container ID: $CONTAINER_ID"

# Get runtime type
RUNTIME_TYPE=$(sudo crictl inspect $CONTAINER_ID | jq -r '.info.runtimeType')
echo "Runtime Type: $RUNTIME_TYPE"

# Create directory
sudo mkdir -p /opt/course/17

# Write to files
echo "$CONTAINER_ID $RUNTIME_TYPE" | sudo tee /opt/course/17/pod-container.txt
sudo crictl logs $CONTAINER_ID | sudo tee /opt/course/17/pod-container.log

# Verify
cat /opt/course/17/pod-container.txt
head -10 /opt/course/17/pod-container.log

ENDSSH
```

Summary of Key Commands

kubectl Common Commands

```
# Context and config
kubectl config get-contexts
kubectl config current-context
kubectl config view --raw

# Resources
kubectl get pods -n <namespace> -o wide --show-labels
kubectl describe pod <pod-name> -n <namespace>
kubectl logs <pod-name> -n <namespace>
kubectl exec -it <pod-name> -n <namespace> -- sh

# Scaling
kubectl scale deployment <name> --replicas=<count> -n <namespace>

# RBAC
kubectl create serviceaccount <name> -n <namespace>
kubectl create role <name> --verb=<verbs> --resource=<resources> -n <namespace>
kubectl create rolebinding <name> --role=<role> --serviceaccount=<namespace>:<sa> -n <namespace>
kubectl auth can-i <verb> <resource> --as=system:serviceaccount:<namespace>:<sa> -n <namespace>

# Certificates
kubeadm certs check-expiration
kubeadm certs renew apiserver
openssl x509 -in <cert> -noout -enddate

# Node management
kubeadm token create --print-join-command
kubectl get nodes
kubectl describe node <node-name>
```

crictl Commands

```
sudo crictl ps                    # List containers
sudo crictl pods                  # List pods
sudo crictl ps --pod=<POD_ID>    # List containers in pod
sudo crictl inspect <CONTAINER_ID> # Inspect container
sudo crictl logs <CONTAINER_ID>   # Get container logs
```

Helm Commands

```
helm repo add <name> <url>
helm repo update
helm install <release-name> <chart> --namespace <namespace>
helm list -n <namespace>
```

Important Concepts

QoS Classes (Pod Eviction Priority)

1. **BestEffort** - No resource requests/limits (evicted first)
2. **Burstable** - Partial requests/limits (evicted second)
3. **Guaranteed** - requests = limits (evicted last)

NetworkPolicy Types

• **Ingress** - Controls incoming traffic
• **Egress** - Controls outgoing traffic
• Default deny when NetworkPolicy is applied

Pod Anti-Affinity

• `requiredDuringSchedulingIgnoredDuringExecution` - Hard requirement
• `preferredDuringSchedulingIgnoredDuringExecution` - Soft preference
• `topologyKey` - Defines scope (e.g., `kubernetes.io/hostname` for per-node)

RBAC Components

• **ServiceAccount** - Identity for Pods
• **Role** - Permissions within namespace
• **ClusterRole** - Permissions cluster-wide
• **RoleBinding** - Links Role to subjects
• **ClusterRoleBinding** - Links ClusterRole to subjects

End of Document