1 CKA

Generated on: 2026-01-13 15:04:23 **Version:** 1.0

1.1 Table of Contents

1.2 Overview

The **Certified Kubernetes Administrator (CKA)** exam certifies that candidates have the skills, knowledge, and competency to perform the responsibilities of Kubernetes administrators.

1.3 Exam Overview

Detail	Information
Exam Format	Performance-based (hands-on)
Number of Questions	15-20
Duration	2 hours
Passing Score	66%
Certification Validity	3 years
Cost	$395 USD
Retake Policy	1 free retake
Kubernetes Version	1.30

1.4 Exam Domains & Weights

Domain	Weight
Cluster Architecture, Installation & Configuration	25%
Workloads & Scheduling	15%
Services & Networking	20%
Storage	10%
Troubleshooting	30%

1.5 Prerequisites

- Strong Linux command line skills
- Understanding of Kubernetes architecture
- Experience with containerization (Docker)
- Basic networking knowledge

1.6 Study Resources

1.6.1 Official Resources

- CKA Exam Curriculum
- Kubernetes Documentation
- Kubernetes Tasks

1.6.2 Recommended Courses

- Kubernetes Fundamentals (LFS258)
- CKA with Practice Tests - Udemy

1.6.3 Practice Resources

- Killercoda CKA Scenarios ⭐ **Highly Recommended**
- killer.sh CKA Simulator - Included with exam registration
- Kubernetes Playground

1.7 Quick Navigation

- 01 - Cluster Architecture, Installation & Configuration
- 02 - Workloads & Scheduling
- 03 - Services & Networking
- 04 - Storage
- 05 - Troubleshooting
- Sample Practice Questions

1.8 Exam Environment

The CKA exam provides:

- Access to multiple Kubernetes clusters
- `kubectl` with auto-completion enabled
- Access to Kubernetes documentation (kubernetes.io)
- A Linux terminal environment
- Root access via `sudo`

1.8.1 Allowed Resources During Exam

- kubernetes.io/docs
- kubernetes.io/blog
- helm.sh/docs

1.9 Exam Tips

1. **Master cluster administration** - Know how to bootstrap clusters with kubeadm
2. **Practice troubleshooting** - 30% of the exam focuses on this
3. **Know etcd backup/restore** - Critical for cluster recovery
4. **Understand RBAC** - Role, ClusterRole, RoleBinding, ClusterRoleBinding
5. **Practice certificate management** - TLS certificates for cluster components
6. **Use aliases** - Set up `alias k=kubectl` and enable auto-completion
7. **Bookmark important docs** - Prepare bookmarks for quick access
8. **Practice on Killercoda** - Free hands-on scenarios

1.10 Useful kubectl Commands

```
# Set alias
alias k=kubectl

# Enable auto-completion
source <(kubectl completion bash)
complete -o default -F __start_kubectl k

# Cluster info
kubectl cluster-info
kubectl get nodes -o wide
kubectl get componentstatuses

# Context management
kubectl config get-contexts
kubectl config use-context <context-name>
kubectl config set-context --current --namespace=<namespace>

# Resource management
kubectl api-resources
kubectl explain pod.spec.containers

# Quick operations
kubectl run nginx --image=nginx --dry-run=client -o yaml
kubectl create deployment nginx --image=nginx --dry-run=client -o
        yaml
kubectl expose deployment nginx --port=80 --type=NodePort

# Troubleshooting
kubectl describe node <node-name>
kubectl logs <pod-name> -c <container-name>
kubectl exec -it <pod-name> -- /bin/sh
kubectl top nodes
kubectl top pods
```

1.11 Cluster Administration Commands

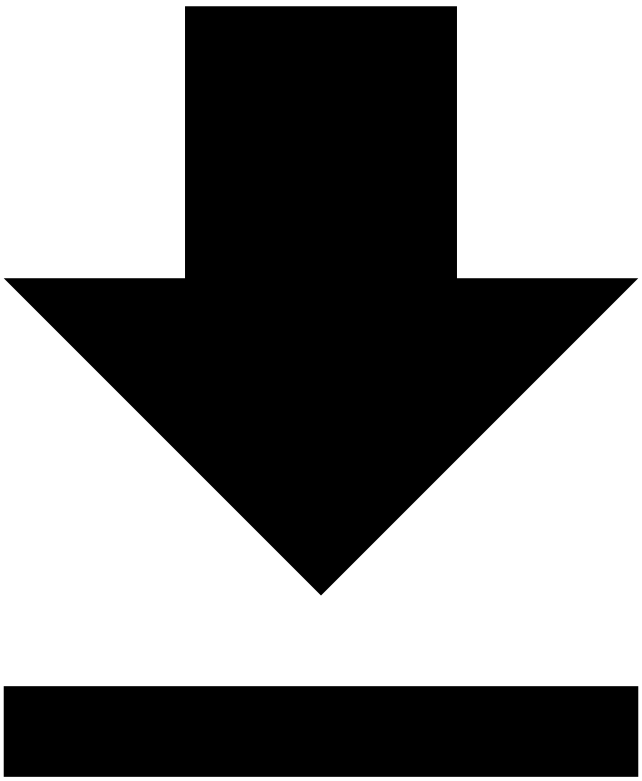
```
# Kubeadm commands
kubeadm init --pod-network-cidr=10.244.0.0/16
kubeadm join <master-ip>:6443 --token <token> --discovery-token-
        ca-cert-hash <hash>
kubeadm token create --print-join-command

# Certificate management
kubeadm certs check-expiration
kubeadm certs renew all

# etcd backup
ETCDCTL_API=3 etcdctl snapshot save /backup/etcd-snapshot.db \
  --endpoints=https://127.0.0.1:2379 \
```

```
  --cacert=/etc/kubernetes/pki/etcd/ca.crt \
  --cert=/etc/kubernetes/pki/etcd/server.crt \
  --key=/etc/kubernetes/pki/etcd/server.key

# etcd restore
ETCDCTL_API=3 etcdctl snapshot restore /backup/etcd-snapshot.db \
  --data-dir=/var/lib/etcd-restore

# Node maintenance
kubectl drain <node-name> --ignore-daemonsets --delete-emptydir-
        data
kubectl cordon <node-name>
kubectl uncordon <node-name>

# Upgrade cluster
kubeadm upgrade plan
kubeadm upgrade apply v1.30.0
```

1.12 Registration

[Register for CKA Exam]

1.13 Cluster Architecture, Installation & Configuration



Download PDF Version

This domain covers Kubernetes cluster architecture, installation, and configuration.

1.14 Kubernetes Architecture

1.14.1 Control Plane Components

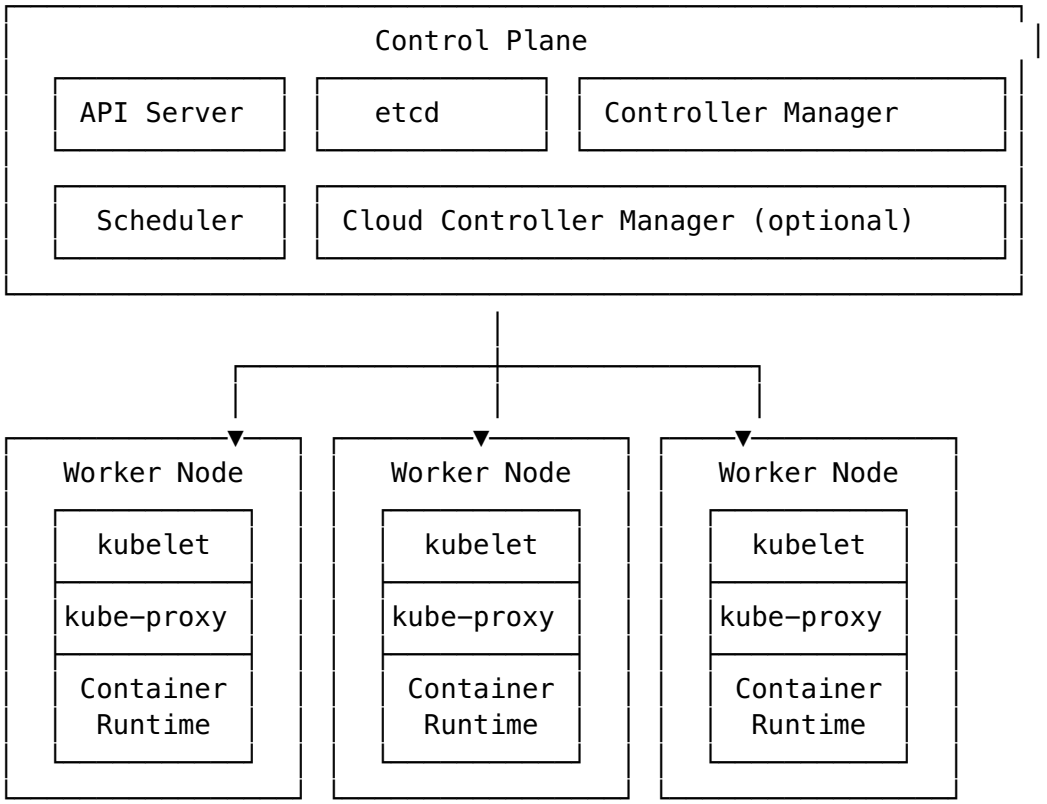
Component	Description
kube-apiserver	Frontend for the Kubernetes control plane
etcd	Consistent and highly-available key-value store
kube-scheduler	Watches for newly created Pods and assigns nodes
kube-controller-manager	Runs controller processes

Component	Description
cloud-controller-manager	Embeds cloud-specific control logic

1.14.2 Node Components

Component	Description
kubelet	Agent that runs on each node
kube-proxy	Network proxy that runs on each node
Container Runtime	Software responsible for running containers

1.14.3 Architecture Diagram



1.15 Cluster Installation with kubeadm

1.15.1 Prerequisites

```
# Disable swap
sudo swapoff -a
sudo sed -i '/ swap / s/^\(.*\)$/#\1/g' /etc/fstab

# Load kernel modules
cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
overlay
br_netfilter
```

```
EOF

sudo modprobe overlay
sudo modprobe br_netfilter

# Set sysctl parameters
cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-iptables  = 1
net.bridge.bridge-nf-call-ip6tables = 1
net.ipv4.ip_forward                 = 1
EOF

sudo sysctl --system
```

1.15.2 Install Container Runtime (containerd)

```
# Install containerd
sudo apt-get update
sudo apt-get install -y containerd

# Configure containerd
sudo mkdir -p /etc/containerd
containerd config default | sudo tee /etc/containerd/config.toml

# Enable SystemdCgroup
sudo sed -i 's/SystemdCgroup = false/SystemdCgroup = true/g' /etc/
        containerd/config.toml

sudo systemctl restart containerd
sudo systemctl enable containerd
```

1.15.3 Install kubeadm, kubelet, kubectl

```
# Add Kubernetes apt repository
sudo apt-get update
sudo apt-get install -y apt-transport-https ca-certificates curl

curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.30/deb/
        Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/
        kubernetes-apt-keyring.gpg

echo
        'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-
        keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.30/
        deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list

# Install packages
sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
```

1.15.4 Initialize Control Plane

```
# Initialize cluster
sudo kubeadm init --pod-network-cidr=10.244.0.0/16

# Configure kubectl for regular user
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config

# Install CNI (Flannel example)
kubectl apply -f https://github.com/flannel-io/flannel/releases/
        latest/download/kube-flannel.yml
```

1.15.5 Join Worker Nodes

```
# On control plane, get join command
kubeadm token create --print-join-command

# On worker node
sudo kubeadm join <control-plane-ip>:6443 --token <token> \
  --discovery-token-ca-cert-hash sha256:<hash>
```

1.16 Cluster Upgrade

1.16.1 Upgrade Control Plane

```
# Check available versions
sudo apt-cache madison kubeadm

# Upgrade kubeadm
sudo apt-mark unhold kubeadm
sudo apt-get update && sudo apt-get install -y kubeadm=1.30.0-1.1
sudo apt-mark hold kubeadm

# Plan upgrade
sudo kubeadm upgrade plan

# Apply upgrade
sudo kubeadm upgrade apply v1.30.0

# Upgrade kubelet and kubectl
sudo apt-mark unhold kubelet kubectl
sudo apt-get update && sudo apt-get install -y kubelet=1.30.0-1.1
        kubectl=1.30.0-1.1
sudo apt-mark hold kubelet kubectl

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

1.16.2 Upgrade Worker Nodes

```
# Drain node (from control plane)
kubectl drain <node-name> --ignore-daemonsets --delete-emptydir-
        data

# On worker node - upgrade kubeadm
sudo apt-mark unhold kubeadm
sudo apt-get update && sudo apt-get install -y kubeadm=1.30.0-1.1
sudo apt-mark hold kubeadm

# Upgrade node config
sudo kubeadm upgrade node

# Upgrade kubelet and kubectl
sudo apt-mark unhold kubelet kubectl
sudo apt-get update && sudo apt-get install -y kubelet=1.30.0-1.1
        kubectl=1.30.0-1.1
sudo apt-mark hold kubelet kubectl

sudo systemctl daemon-reload
sudo systemctl restart kubelet

# Uncordon node (from control plane)
kubectl uncordon <node-name>
```

1.17 etcd Backup and Restore

1.17.1 Backup etcd

```
# Using etcdctl
ETCDCTL_API=3 etcdctl snapshot save /backup/etcd-snapshot.db \
  --endpoints=https://127.0.0.1:2379 \
  --cacert=/etc/kubernetes/pki/etcd/ca.crt \
  --cert=/etc/kubernetes/pki/etcd/server.crt \
  --key=/etc/kubernetes/pki/etcd/server.key

# Verify backup
ETCDCTL_API=3 etcdctl snapshot status /backup/etcd-snapshot.db --
        write-out=table
```

1.17.2 Restore etcd

```
# Stop kube-apiserver (if running as static pod, move manifest)
sudo mv /etc/kubernetes/manifests/kube-apiserver.yaml /tmp/

# Restore snapshot
ETCDCTL_API=3 etcdctl snapshot restore /backup/etcd-snapshot.db \
  --data-dir=/var/lib/etcd-restore

# Update etcd manifest to use new data directory
```

```
# Edit /etc/kubernetes/manifests/etcd.yaml
# Change: --data-dir=/var/lib/etcd-restore

# Restore kube-apiserver
sudo mv /tmp/kube-apiserver.yaml /etc/kubernetes/manifests/
```

1.18 RBAC (Role-Based Access Control)

1.18.1 Role and RoleBinding (Namespace-scoped)

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: default
  name: pod-reader
rules:
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["get", "watch", "list"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: read-pods
  namespace: default
subjects:
- kind: User
  name: jane
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: pod-reader
  apiGroup: rbac.authorization.k8s.io
```

1.18.2 ClusterRole and ClusterRoleBinding (Cluster-scoped)

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: secret-reader
rules:
- apiGroups: [""]
  resources: ["secrets"]
  verbs: ["get", "watch", "list"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: read-secrets-global
subjects:
```

```yaml
- kind: Group
  name: managers
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: secret-reader
  apiGroup: rbac.authorization.k8s.io
```

1.18.3 RBAC Commands

```
# Create role
kubectl create role pod-reader --verb=get,list,watch --
        resource=pods

# Create rolebinding
kubectl create rolebinding read-pods --role=pod-reader --user=jane

# Create clusterrole
kubectl create clusterrole node-reader --verb=get,list,watch --
        resource=nodes

# Create clusterrolebinding
kubectl create clusterrolebinding read-nodes --clusterrole=node-
        reader --user=jane

# Check permissions
kubectl auth can-i create pods --as jane
kubectl auth can-i list nodes --as jane
kubectl auth can-i --list --as jane
```

1.19 Certificates

1.19.1 View Certificate Details

```
# View certificate
openssl x509 -in /etc/kubernetes/pki/apiserver.crt -text -noout

# Check certificate expiration
kubeadm certs check-expiration

# Renew certificates
kubeadm certs renew all
```

1.19.2 Certificate Locations

Certificate	Path
CA	`/etc/kubernetes/pki/ca.crt`
API Server	`/etc/kubernetes/pki/apiserver.crt`
API Server Key	`/etc/kubernetes/pki/apiserver.key`

Certificate	Path
etcd CA	`/etc/kubernetes/pki/etcd/ca.crt`
etcd Server	`/etc/kubernetes/pki/etcd/server.crt`

1.20 Key Concepts to Remember

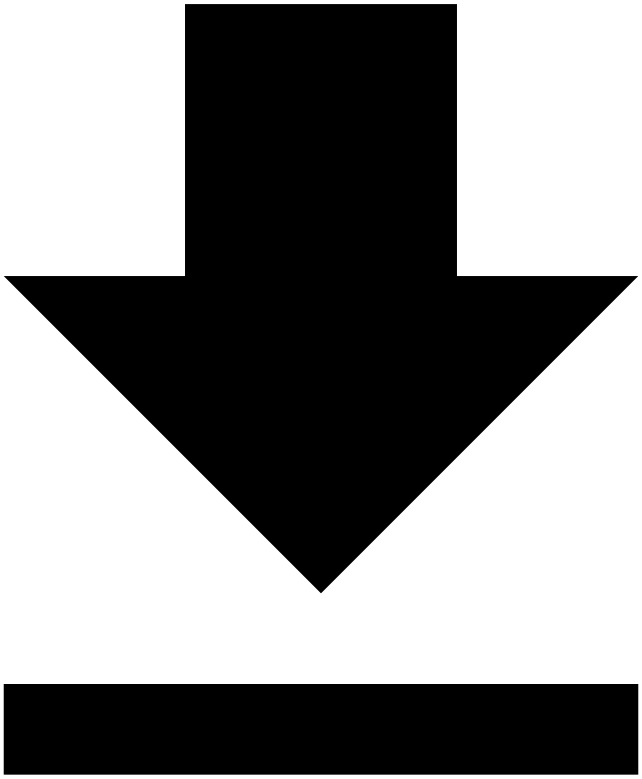
1. **Control Plane** - API Server, etcd, Scheduler, Controller Manager
2. **kubeadm** - Tool to bootstrap clusters
3. **etcd** - Cluster state storage, backup/restore critical
4. **RBAC** - Role, ClusterRole, RoleBinding, ClusterRoleBinding
5. **Certificates** - TLS for secure communication

1.21 Practice Questions

1. How do you initialize a Kubernetes cluster with kubeadm?
2. What is the command to backup etcd?
3. How do you upgrade a cluster from 1.29 to 1.30?
4. What is the difference between Role and ClusterRole?
5. How do you check if a user can perform an action?

Back to CKA Overview | Next: Workloads & Scheduling →

1.22 Workloads & Scheduling



Download PDF Version

This domain covers managing workloads and understanding Kubernetes scheduling.

1.23 Deployments

1.23.1 Creating Deployments

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
```

```yaml
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 1
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.21
        ports:
        - containerPort: 80
        resources:
          requests:
            memory: "64Mi"
            cpu: "250m"
          limits:
            memory: "128Mi"
            cpu: "500m"
```

1.23.2 Deployment Commands

```bash
# Create deployment
kubectl create deployment nginx --image=nginx:1.21 --replicas=3

# Scale deployment
kubectl scale deployment nginx --replicas=5

# Update image
kubectl set image deployment/nginx nginx=nginx:1.22

# Rollout commands
kubectl rollout status deployment/nginx
kubectl rollout history deployment/nginx
kubectl rollout undo deployment/nginx
kubectl rollout undo deployment/nginx --to-revision=2
kubectl rollout pause deployment/nginx
kubectl rollout resume deployment/nginx
```

1.24 DaemonSets

Ensures all (or some) nodes run a copy of a Pod.

```yaml
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: fluentd
  namespace: kube-system
```

```
spec:
  selector:
    matchLabels:
      name: fluentd
  template:
    metadata:
      labels:
        name: fluentd
    spec:
      tolerations:
      - key: node-role.kubernetes.io/control-plane
        operator: Exists
        effect: NoSchedule
      containers:
      - name: fluentd
        image: fluentd:v1.14
        volumeMounts:
        - name: varlog
          mountPath: /var/log
      volumes:
      - name: varlog
        hostPath:
          path: /var/log
```

1.25 StatefulSets

For stateful applications requiring stable network identities and persistent storage.

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: mysql
spec:
  serviceName: mysql
  replicas: 3
  selector:
    matchLabels:
      app: mysql
  template:
    metadata:
      labels:
        app: mysql
    spec:
      containers:
      - name: mysql
        image: mysql:8.0
        ports:
        - containerPort: 3306
        volumeMounts:
        - name: data
          mountPath: /var/lib/mysql
```

```
        env:
        - name: MYSQL_ROOT_PASSWORD
          valueFrom:
            secretKeyRef:
              name: mysql-secret
              key: password
  volumeClaimTemplates:
  - metadata:
      name: data
    spec:
      accessModes: ["ReadWriteOnce"]
      resources:
        requests:
          storage: 10Gi
```

1.26 Scheduling

1.26.1 Node Selectors

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  nodeSelector:
    disktype: ssd
  containers:
  - name: nginx
    image: nginx
```

1.26.2 Node Affinity

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: topology.kubernetes.io/zone
            operator: In
            values:
            - us-east-1a
            - us-east-1b
      preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 1
        preference:
```

```yaml
          matchExpressions:
          - key: disktype
            operator: In
            values:
            - ssd
  containers:
  - name: nginx
    image: nginx
```

1.26.3 Pod Affinity and Anti-Affinity

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: web-server
spec:
  affinity:
    podAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
      - labelSelector:
          matchExpressions:
          - key: app
            operator: In
            values:
            - cache
        topologyKey: kubernetes.io/hostname
    podAntiAffinity:
      preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 100
        podAffinityTerm:
          labelSelector:
            matchExpressions:
            - key: app
              operator: In
              values:
              - web-server
          topologyKey: kubernetes.io/hostname
  containers:
  - name: web
    image: nginx
```

1.26.4 Taints and Tolerations

```bash
# Add taint to node
kubectl taint nodes node1 key=value:NoSchedule
kubectl taint nodes node1 key=value:NoExecute
kubectl taint nodes node1 key=value:PreferNoSchedule

# Remove taint
kubectl taint nodes node1 key=value:NoSchedule-
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  tolerations:
  - key: "key"
    operator: "Equal"
    value: "value"
    effect: "NoSchedule"
  - key: "key"
    operator: "Exists"
    effect: "NoExecute"
    tolerationSeconds: 3600
  containers:
  - name: nginx
    image: nginx
```

1.26.5 Taint Effects

Effect	Description
`NoSchedule`	New pods won't be scheduled
`PreferNoSchedule`	Scheduler tries to avoid
`NoExecute`	Evicts existing pods

1.27 Resource Management

1.27.1 Resource Requests and Limits

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: resource-demo
spec:
  containers:
  - name: app
    image: nginx
    resources:
      requests:
        memory: "64Mi"
        cpu: "250m"
      limits:
        memory: "128Mi"
        cpu: "500m"
```

1.27.2 LimitRange

```yaml
apiVersion: v1
kind: LimitRange
metadata:
  name: resource-limits
  namespace: default
spec:
  limits:
  - type: Container
    default:
      cpu: "500m"
      memory: "256Mi"
    defaultRequest:
      cpu: "100m"
      memory: "128Mi"
    max:
      cpu: "2"
      memory: "1Gi"
    min:
      cpu: "50m"
      memory: "64Mi"
```

1.27.3 ResourceQuota

```yaml
apiVersion: v1
kind: ResourceQuota
metadata:
  name: compute-quota
  namespace: default
spec:
  hard:
    requests.cpu: "4"
    requests.memory: "8Gi"
    limits.cpu: "8"
    limits.memory: "16Gi"
    pods: "10"
    configmaps: "10"
    secrets: "10"
```

1.28 Static Pods

Static pods are managed directly by kubelet on a specific node.

```
# Default static pod path
/etc/kubernetes/manifests/

# Check kubelet config for static pod path
cat /var/lib/kubelet/config.yaml | grep staticPodPath

# Create static pod
```

```
cat <<EOF > /etc/kubernetes/manifests/static-nginx.yaml
apiVersion: v1
kind: Pod
metadata:
  name: static-nginx
spec:
  containers:
  - name: nginx
    image: nginx
EOF
```

1.29 Multiple Schedulers

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: custom-scheduler-pod
spec:
  schedulerName: my-custom-scheduler
  containers:
  - name: nginx
    image: nginx
```

1.30 Manual Scheduling

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: manual-pod
spec:
  nodeName: node01  # Bypass scheduler
  containers:
  - name: nginx
    image: nginx
```

1.31 Labels and Selectors

```
# Add label
kubectl label nodes node1 disktype=ssd
kubectl label pods nginx env=prod

# Remove label
kubectl label nodes node1 disktype-

# Show labels
kubectl get nodes --show-labels
kubectl get pods -l env=prod
kubectl get pods -l 'env in (prod,dev)'
```

1.32 Key Concepts to Remember

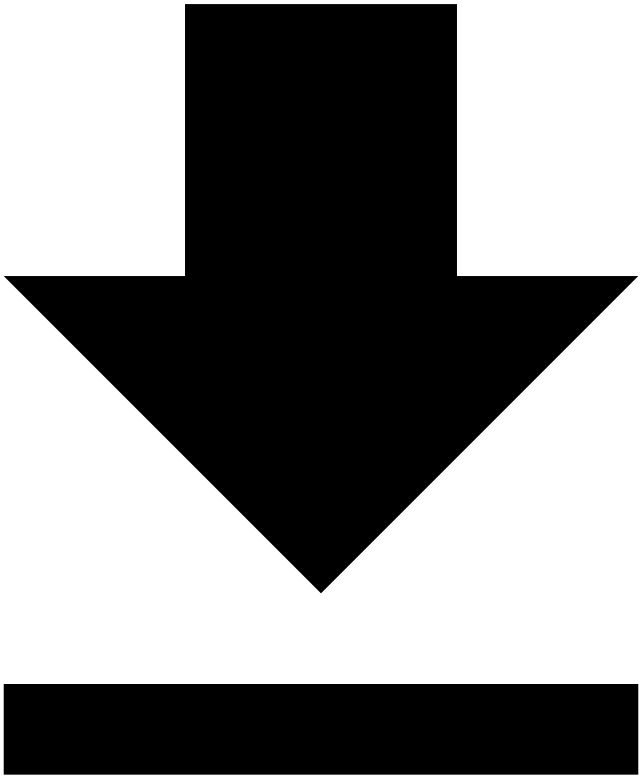
1. **Deployments** - Declarative updates for Pods
2. **DaemonSets** - One pod per node
3. **StatefulSets** - Stable network identity and storage
4. **Node Affinity** - Schedule based on node labels
5. **Taints/Tolerations** - Repel pods from nodes
6. **Static Pods** - Managed by kubelet directly

1.33 Practice Questions

1. How do you create a DaemonSet that runs on all nodes including control plane?
2. What is the difference between nodeSelector and nodeAffinity?
3. How do you taint a node to prevent scheduling?
4. Where are static pod manifests stored?
5. How do you schedule a pod to a specific node?

← Previous: Cluster Architecture | Back to CKA Overview | Next: Services & Networking →

1.34 Services & Networking



Download PDF Version

This domain covers Kubernetes networking concepts, Services, DNS, and network policies.

1.35 Services

1.35.1 Service Types

Type	Description
`ClusterIP`	Internal cluster IP (default)
`NodePort`	Exposes on each node's IP at a static port (30000-32767)
`LoadBalancer`	External load balancer (cloud provider)
`ExternalName`	Maps to external DNS name

1.35.2 ClusterIP Service

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  type: ClusterIP
  selector:
    app: myapp
  ports:
  - port: 80
    targetPort: 8080
    protocol: TCP
```

1.35.3 NodePort Service

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-nodeport
spec:
  type: NodePort
  selector:
    app: myapp
  ports:
  - port: 80
    targetPort: 8080
    nodePort: 30080
```

1.35.4 LoadBalancer Service

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-loadbalancer
spec:
  type: LoadBalancer
  selector:
    app: myapp
  ports:
  - port: 80
    targetPort: 8080
```

1.35.5 Headless Service

```yaml
apiVersion: v1
kind: Service
metadata:
  name: headless-service
spec:
```

```
  clusterIP: None
  selector:
    app: myapp
  ports:
  - port: 80
    targetPort: 8080
```

1.35.6 Service Commands

```
# Create service
kubectl expose deployment nginx --port=80 --target-port=8080 --
        type=ClusterIP

# Create NodePort service
kubectl expose deployment nginx --port=80 --type=NodePort

# Get endpoints
kubectl get endpoints my-service
```

1.36 DNS in Kubernetes

1.36.1 Service DNS

```
<service-name>.<namespace>.svc.cluster.local

Examples:
- my-service.default.svc.cluster.local
- my-service.default.svc
- my-service.default
- my-service (within same namespace)
```

1.36.2 Pod DNS

```
<pod-ip-dashed>.<namespace>.pod.cluster.local

Example:
- 10-244-0-5.default.pod.cluster.local
```

1.36.3 CoreDNS

```
# Check CoreDNS pods
kubectl get pods -n kube-system -l k8s-app=kube-dns

# Check CoreDNS ConfigMap
kubectl get configmap coredns -n kube-system -o yaml

# Test DNS resolution
kubectl run test --image=busybox:1.36 --rm -it -- nslookup
        kubernetes
```

1.37 Ingress

1.37.1 Ingress Resource

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: my-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  ingressClassName: nginx
  rules:
  - host: myapp.example.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: myapp-service
            port:
              number: 80
```

1.37.2 Path-based Routing

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: path-ingress
spec:
  ingressClassName: nginx
  rules:
  - host: myapp.example.com
    http:
      paths:
      - path: /api
        pathType: Prefix
        backend:
          service:
            name: api-service
            port:
              number: 80
      - path: /web
        pathType: Prefix
        backend:
          service:
            name: web-service
            port:
              number: 80
```

1.37.3 TLS Ingress

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: tls-ingress
spec:
  ingressClassName: nginx
  tls:
  - hosts:
    - myapp.example.com
    secretName: tls-secret
  rules:
  - host: myapp.example.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: myapp-service
            port:
              number: 80
```

1.37.4 Ingress Commands

```bash
# Create ingress
kubectl create ingress my-ingress \
  --rule="myapp.example.com/=myapp-service:80" \
  --class=nginx

# With TLS
kubectl create ingress my-ingress \
  --rule="myapp.example.com/=myapp-service:80,tls=tls-secret" \
  --class=nginx
```

1.38 Network Policies

1.38.1 Default Deny All Ingress

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

1.38.2 Default Deny All Egress

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: deny-all-egress
  namespace: default
spec:
  podSelector: {}
  policyTypes:
  - Egress
```

1.38.3 Allow Specific Ingress

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-frontend
  namespace: default
spec:
  podSelector:
    matchLabels:
      app: backend
  policyTypes:
  - Ingress
  ingress:
  - from:
    - podSelector:
        matchLabels:
          app: frontend
    - namespaceSelector:
        matchLabels:
          name: production
    ports:
    - protocol: TCP
      port: 8080
```

1.38.4 Allow Egress to Specific Pods and DNS

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-egress
  namespace: default
spec:
  podSelector:
    matchLabels:
      app: backend
  policyTypes:
  - Egress
  egress:
```

```
  - to:
    - podSelector:
        matchLabels:
          app: database
    ports:
    - protocol: TCP
      port: 5432
  - to:  # Allow DNS
    ports:
    - protocol: UDP
      port: 53
```

1.38.5 IP Block

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-external
spec:
  podSelector:
    matchLabels:
      app: web
  policyTypes:
  - Ingress
  ingress:
  - from:
    - ipBlock:
        cidr: 10.0.0.0/8
        except:
        - 10.0.1.0/24
```

1.39 CNI (Container Network Interface)

1.39.1 Common CNI Plugins

Plugin	Description
Flannel	Simple overlay network
Calico	Network policy support, BGP
Weave	Mesh network
Cilium	eBPF-based networking

1.39.2 CNI Configuration

```
# CNI config location
/etc/cni/net.d/

# CNI binaries
/opt/cni/bin/
```

```
# Check CNI plugin
ls /etc/cni/net.d/
cat /etc/cni/net.d/10-flannel.conflist
```

1.40 Cluster Networking

1.40.1 Pod Networking

```
# View pod IPs
kubectl get pods -o wide

# Check pod network
kubectl exec -it nginx -- ip addr
kubectl exec -it nginx -- ip route
```

1.40.2 Service Networking

```
# Check service CIDR
kubectl cluster-info dump | grep -m 1 service-cluster-ip-range

# Check kube-proxy mode
kubectl logs -n kube-system -l k8s-app=kube-proxy | grep "Using"
```

1.40.3 Port Forwarding

```
# Forward pod port
kubectl port-forward pod/nginx 8080:80

# Forward service port
kubectl port-forward svc/nginx 8080:80

# Forward to all interfaces
kubectl port-forward --address 0.0.0.0 pod/nginx 8080:80
```

1.41 Key Concepts to Remember

1. **ClusterIP** - Default, internal only
2. **NodePort** - External access via node IP:port (30000-32767)
3. **LoadBalancer** - Cloud provider load balancer
4. **Ingress** - HTTP/HTTPS routing, requires controller
5. **Network Policies** - Default allow, explicit deny
6. **CoreDNS** - Cluster DNS service

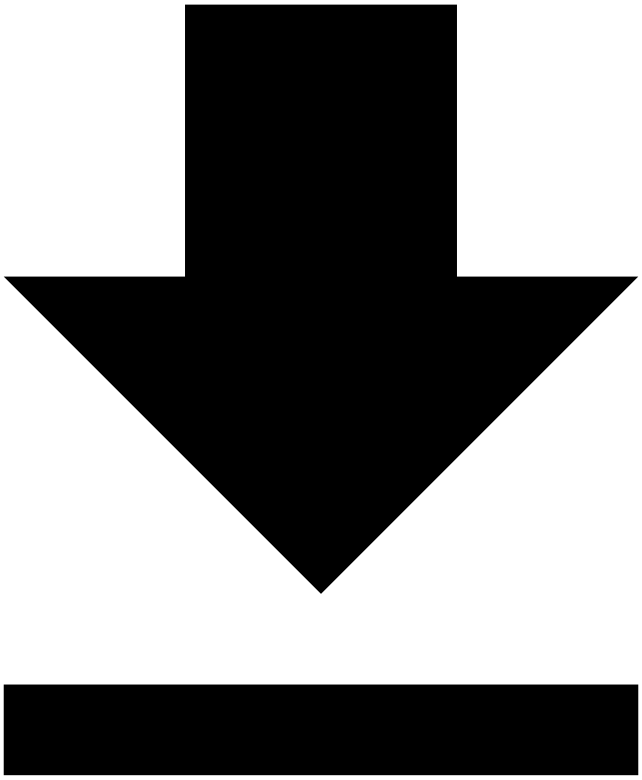
1.42 Practice Questions

1. What is the default Service type in Kubernetes?

2. How do you create a Service that exposes a deployment externally?
3. What is the DNS name format for a Service?
4. How do you create a NetworkPolicy that denies all ingress traffic?
5. What is the difference between Ingress and LoadBalancer Service?

← Previous: Workloads & Scheduling | Back to CKA Overview | Next: Storage →

1.43 Storage



Download PDF Version

This domain covers Kubernetes storage concepts including volumes, persistent volumes, and storage classes.

1.44 Volumes

1.44.1 emptyDir

Temporary storage that exists for the Pod's lifetime.

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod-with-emptydir
spec:
  containers:
  - name: app
    image: nginx
    volumeMounts:
    - name: cache
      mountPath: /cache
  volumes:
  - name: cache
    emptyDir: {}
```

1.44.2 Memory-backed emptyDir

```yaml
volumes:
- name: memory-cache
  emptyDir:
    medium: Memory
    sizeLimit: 100Mi
```

1.44.3 hostPath

Mount a file or directory from the host node.

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod-with-hostpath
spec:
  containers:
  - name: app
    image: nginx
    volumeMounts:
    - name: data
      mountPath: /data
  volumes:
  - name: data
    hostPath:
      path: /data
      type: DirectoryOrCreate
```

1.44.4 hostPath Types

Type	Description
`""`	No checks (default)
`DirectoryOrCreate`	Create directory if not exists
`Directory`	Directory must exist
`FileOrCreate`	Create file if not exists
`File`	File must exist
`Socket`	Unix socket must exist

1.45 Persistent Volumes (PV)

1.45.1 PersistentVolume

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-volume
spec:
  capacity:
    storage: 10Gi
  accessModes:
  - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  storageClassName: manual
  hostPath:
    path: /mnt/data
```

1.45.2 Access Modes

Mode	Abbreviation	Description
`ReadWriteOnce`	RWO	Single node read-write
`ReadOnlyMany`	ROX	Multiple nodes read-only
`ReadWriteMany`	RWX	Multiple nodes read-write
`ReadWriteOncePod`	RWOP	Single pod read-write

1.45.3 Reclaim Policies

Policy	Description
`Retain`	Manual reclamation
`Delete`	Delete volume when PVC is deleted
`Recycle`	Basic scrub (deprecated)

1.46 PersistentVolumeClaim (PVC)

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-claim
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 5Gi
  storageClassName: manual
```

1.46.1 Using PVC in Pod

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod-with-pvc
spec:
  containers:
  - name: app
    image: nginx
    volumeMounts:
    - name: data
      mountPath: /data
  volumes:
  - name: data
    persistentVolumeClaim:
      claimName: pvc-claim
```

1.47 Storage Classes

1.47.1 StorageClass

```yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: fast
provisioner: kubernetes.io/aws-ebs
parameters:
  type: gp3
  iopsPerGB: "10"
reclaimPolicy: Delete
allowVolumeExpansion: true
volumeBindingMode: WaitForFirstConsumer
```

1.47.2 Volume Binding Modes

Mode	Description
`Immediate`	Bind PV immediately when PVC is created
`WaitForFirstConsumer`	Delay binding until Pod is scheduled

1.47.3 Dynamic Provisioning

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: dynamic-pvc
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
  storageClassName: fast  # Uses StorageClass for dynamic
      provisioning
```

1.48 ConfigMap as Volume

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod-with-configmap
spec:
  containers:
  - name: app
    image: nginx
    volumeMounts:
    - name: config
      mountPath: /etc/config
  volumes:
  - name: config
    configMap:
      name: my-config
      items:
      - key: config.json
        path: app-config.json
```

1.49 Secret as Volume

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod-with-secret
```

```yaml
spec:
  containers:
  - name: app
    image: nginx
    volumeMounts:
    - name: secret
      mountPath: /etc/secrets
      readOnly: true
  volumes:
  - name: secret
    secret:
      secretName: my-secret
      defaultMode: 0400
```

1.50 Projected Volumes

Combine multiple volume sources into a single directory.

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod-with-projected
spec:
  containers:
  - name: app
    image: nginx
    volumeMounts:
    - name: all-in-one
      mountPath: /projected-volume
  volumes:
  - name: all-in-one
    projected:
      sources:
      - secret:
          name: my-secret
      - configMap:
          name: my-config
      - downwardAPI:
          items:
          - path: labels
            fieldRef:
              fieldPath: metadata.labels
```

1.51 Volume Expansion

```yaml
# StorageClass must have allowVolumeExpansion: true
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: expandable-pvc
```

```yaml
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 20Gi  # Increase from original size
  storageClassName: fast
```

1.52 Storage Commands

```bash
# List PersistentVolumes
kubectl get pv

# List PersistentVolumeClaims
kubectl get pvc

# List StorageClasses
kubectl get sc

# Describe PV
kubectl describe pv pv-volume

# Delete PVC
kubectl delete pvc pvc-claim

# Patch PVC to expand
kubectl patch pvc pvc-claim -p '{"spec":{"resources":{"requests":
        {"storage":"20Gi"}}}}'
```

1.53 Volume Snapshots

1.53.1 VolumeSnapshotClass

```yaml
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotClass
metadata:
  name: csi-snapclass
driver: ebs.csi.aws.com
deletionPolicy: Delete
```

1.53.2 VolumeSnapshot

```yaml
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
  name: my-snapshot
spec:
  volumeSnapshotClassName: csi-snapclass
```

```
  source:
    persistentVolumeClaimName: my-pvc
```

1.53.3 Restore from Snapshot

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: restored-pvc
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
  dataSource:
    name: my-snapshot
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
```

1.54 Key Concepts to Remember

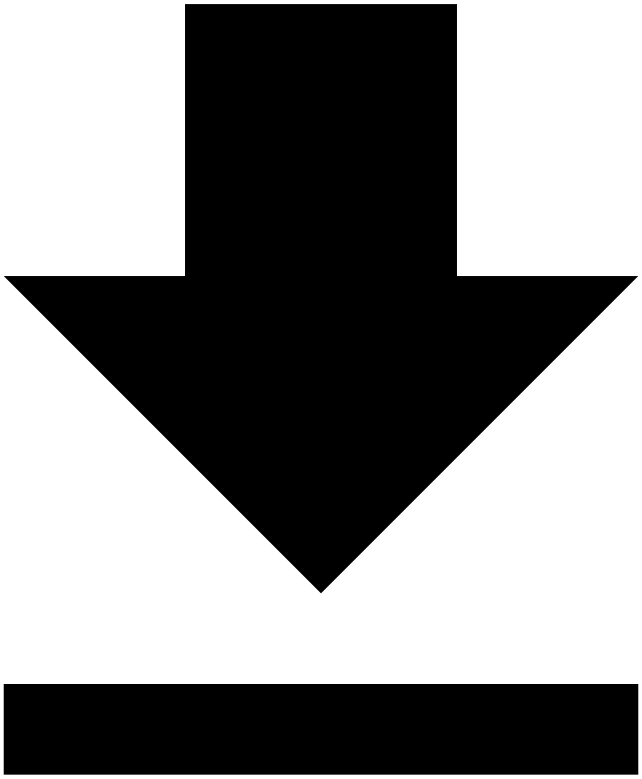
1. **PV** - Cluster-level storage resource
2. **PVC** - Request for storage by a user
3. **StorageClass** - Dynamic provisioning template
4. **Access Modes** - RWO, ROX, RWX, RWOP
5. **Reclaim Policies** - Retain, Delete, Recycle

1.55 Practice Questions

1. What is the difference between PV and PVC?
2. How do you create a PVC that uses dynamic provisioning?
3. What access mode allows multiple nodes to read and write?
4. How do you expand a PVC?
5. What is the purpose of volumeBindingMode: WaitForFirstConsumer?

← Previous: Services & Networking | Back to CKA Overview | Next: Troubleshooting →

1.56 Troubleshooting



Download PDF Version

This domain covers troubleshooting Kubernetes clusters, applications, and networking issues. This is the largest domain in the CKA exam.

1.57 Cluster Troubleshooting

1.57.1 Check Cluster Health

```
# Cluster info
kubectl cluster-info
kubectl cluster-info dump

# Component status (deprecated but useful)
kubectl get componentstatuses

# Check nodes
kubectl get nodes
kubectl describe node <node-name>
```

```
# Check system pods
kubectl get pods -n kube-system
```

1.57.2 Control Plane Components

```
# Check control plane pods (if using kubeadm)
kubectl get pods -n kube-system

# Check static pod manifests
ls /etc/kubernetes/manifests/
cat /etc/kubernetes/manifests/kube-apiserver.yaml

# Check component logs
kubectl logs -n kube-system kube-apiserver-<node>
kubectl logs -n kube-system kube-controller-manager-<node>
kubectl logs -n kube-system kube-scheduler-<node>
kubectl logs -n kube-system etcd-<node>

# If running as systemd services
sudo journalctl -u kubelet
sudo journalctl -u kube-apiserver
```

1.57.3 kubelet Troubleshooting

```
# Check kubelet status
sudo systemctl status kubelet
sudo systemctl restart kubelet

# Check kubelet logs
sudo journalctl -u kubelet -f
sudo journalctl -u kubelet --since "10 minutes ago"

# Check kubelet config
cat /var/lib/kubelet/config.yaml
cat /etc/kubernetes/kubelet.conf
```

1.57.4 etcd Troubleshooting

```
# Check etcd health
ETCDCTL_API=3 etcdctl endpoint health \
  --endpoints=https://127.0.0.1:2379 \
  --cacert=/etc/kubernetes/pki/etcd/ca.crt \
  --cert=/etc/kubernetes/pki/etcd/server.crt \
  --key=/etc/kubernetes/pki/etcd/server.key

# Check etcd members
ETCDCTL_API=3 etcdctl member list \
  --endpoints=https://127.0.0.1:2379 \
  --cacert=/etc/kubernetes/pki/etcd/ca.crt \
```

```
  --cert=/etc/kubernetes/pki/etcd/server.crt \
  --key=/etc/kubernetes/pki/etcd/server.key
```

1.58 Node Troubleshooting

1.58.1 Node Status

```
# Get node details
kubectl get nodes -o wide
kubectl describe node <node-name>

# Check node conditions
kubectl get nodes -o jsonpath='{.items[*].status.conditions}'
```

1.58.2 Node Conditions

Condition	Description
`Ready`	Node is healthy and ready
`MemoryPressure`	Node memory is low
`DiskPressure`	Node disk space is low
`PIDPressure`	Too many processes
`NetworkUnavailable`	Network not configured

1.58.3 Node Maintenance

```
# Cordon node (prevent scheduling)
kubectl cordon <node-name>

# Drain node (evict pods)
kubectl drain <node-name> --ignore-daemonsets --delete-emptydir-
        data

# Uncordon node
kubectl uncordon <node-name>
```

1.59 Application Troubleshooting

1.59.1 Pod Debugging

```
# Get pod status
kubectl get pods
kubectl get pods -o wide
kubectl get pods --all-namespaces

# Describe pod (events, conditions)
kubectl describe pod <pod-name>
```

```
# Get pod YAML
kubectl get pod <pod-name> -o yaml

# Check pod logs
kubectl logs <pod-name>
kubectl logs <pod-name> -c <container-name>
kubectl logs <pod-name> --previous
kubectl logs <pod-name> -f
kubectl logs <pod-name> --tail=100

# Execute command in pod
kubectl exec -it <pod-name> -- /bin/sh
kubectl exec <pod-name> -- cat /etc/config/app.conf
```

1.59.2 Common Pod Issues

Status	Cause	Solution
Pending	No node available, resource constraints	Check events, node resources
ImagePullBackOff	Image not found, auth issues	Check image name, pull secrets
CrashLoopBackOff	Container crashes repeatedly	Check logs, probe config
CreateContainerConfigError	ConfigMap/Secret missing	Check references
OOMKilled	Out of memory	Increase memory limits
Evicted	Node resource pressure	Check node conditions

1.59.3 Debug with Ephemeral Containers

```
# Add debug container to running pod
kubectl debug <pod-name> -it --image=busybox --target=<container-
        name>

# Debug node
kubectl debug node/<node-name> -it --image=ubuntu
```

1.59.4 Pod Resource Issues

```
# Check resource usage
kubectl top pods
kubectl top pods --containers
kubectl top nodes
```

```
# Check resource requests/limits
kubectl describe pod <pod-name> | grep -A 5 "Requests\|Limits"
```

1.60 Service Troubleshooting

1.60.1 Service Debugging

```
# Check service
kubectl get svc
kubectl describe svc <service-name>

# Check endpoints
kubectl get endpoints <service-name>

# Test service from within cluster
kubectl run test --image=busybox:1.36 --rm -it -- wget -qO-
        http://<service-name>

# Check service DNS
kubectl run test --image=busybox:1.36 --rm -it -- nslookup
        <service-name>
```

1.60.2 Common Service Issues

Issue	Cause	Solution
No endpoints	Selector mismatch	Check pod labels match service selector
Connection refused	Wrong port	Check targetPort matches container port
DNS not resolving	CoreDNS issues	Check CoreDNS pods

1.61 Networking Troubleshooting

1.61.1 DNS Debugging

```
# Check CoreDNS
kubectl get pods -n kube-system -l k8s-app=kube-dns
kubectl logs -n kube-system -l k8s-app=kube-dns

# Test DNS resolution
kubectl run test --image=busybox:1.36 --rm -it -- nslookup
        kubernetes
kubectl run test --image=busybox:1.36 --rm -it -- nslookup
        <service>.<namespace>.svc.cluster.local

# Check resolv.conf in pod
kubectl exec <pod-name> -- cat /etc/resolv.conf
```

1.61.2 Network Policy Debugging

```
# List network policies
kubectl get networkpolicies
kubectl describe networkpolicy <policy-name>

# Test connectivity
kubectl exec <pod-name> -- nc -zv <target-ip> <port>
kubectl exec <pod-name> -- wget -qO- --timeout=2 http://<service>
```

1.61.3 CNI Troubleshooting

```
# Check CNI config
ls /etc/cni/net.d/
cat /etc/cni/net.d/*.conf

# Check CNI binaries
ls /opt/cni/bin/

# Check pod networking
kubectl exec <pod-name> -- ip addr
kubectl exec <pod-name> -- ip route
```

1.62 Certificate Troubleshooting

```
# Check certificate expiration
kubeadm certs check-expiration

# View certificate details
openssl x509 -in /etc/kubernetes/pki/apiserver.crt -text -noout

# Check certificate dates
openssl x509 -in /etc/kubernetes/pki/apiserver.crt -noout -dates

# Renew certificates
kubeadm certs renew all
```

1.63 Logging

1.63.1 Container Logs

```
# View logs
kubectl logs <pod-name>
kubectl logs <pod-name> -c <container>
kubectl logs <pod-name> --all-containers

# Follow logs
kubectl logs -f <pod-name>
```

```
# Previous container logs
kubectl logs <pod-name> --previous

# Logs since time
kubectl logs <pod-name> --since=1h
kubectl logs <pod-name> --since-time=2024-01-01T00:00:00Z

# Logs with timestamps
kubectl logs <pod-name> --timestamps
```

1.63.2 System Logs

```
# kubelet logs
sudo journalctl -u kubelet

# Container runtime logs
sudo journalctl -u containerd
sudo journalctl -u docker

# System messages
sudo tail -f /var/log/syslog
sudo tail -f /var/log/messages
```

1.64 Events

```
# Get events
kubectl get events
kubectl get events --sort-by='.lastTimestamp'
kubectl get events -n <namespace>

# Watch events
kubectl get events -w

# Filter events
kubectl get events --field-selector type=Warning
kubectl get events --field-selector involvedObject.name=<pod-name>
```

1.65 Troubleshooting Checklist

1.65.1 Pod Not Starting

1. Check pod status: `kubectl get pod <pod>`
2. Check events: `kubectl describe pod <pod>`
3. Check logs: `kubectl logs <pod>`
4. Check node resources: `kubectl describe node <node>`
5. Check image: `kubectl get pod <pod> -o yaml | grep image`

1.65.2 Service Not Working

1. Check service: `kubectl get svc <service>`
2. Check endpoints: `kubectl get endpoints <service>`
3. Check pod labels match selector
4. Test from within cluster
5. Check network policies

1.65.3 Node Not Ready

1. Check node status: `kubectl describe node <node>`
2. Check kubelet: `systemctl status kubelet`
3. Check kubelet logs: `journalctl -u kubelet`
4. Check container runtime
5. Check disk/memory pressure

1.66 Key Concepts to Remember

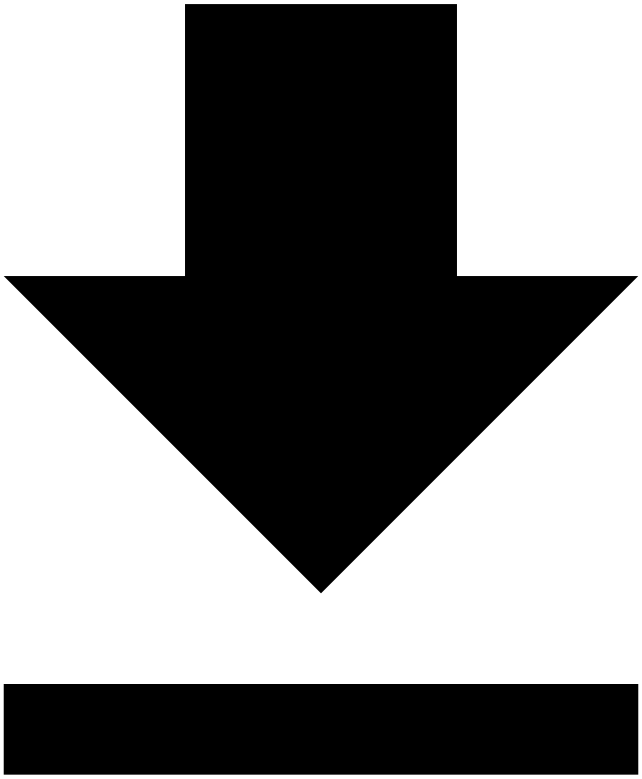
1. **kubectl describe** - First step for troubleshooting
2. **kubectl logs** - Check container output
3. **kubectl exec** - Debug inside container
4. **Events** - Show what happened
5. **journalctl** - System service logs

1.67 Practice Questions

1. A pod is in CrashLoopBackOff status. How do you troubleshoot?
2. How do you check why a node is NotReady?
3. A service has no endpoints. What could be wrong?
4. How do you view kubelet logs?
5. How do you test DNS resolution from within a pod?

← Previous: Storage | Back to CKA Overview | Next: Sample Practice Questions →

1.68 Sample Practice Questions



Download PDF Version

> **Disclaimer**: These are sample practice questions created for study purposes only. They are NOT actual exam questions and are designed to help you test your understanding of CKA concepts. Real exam questions may differ in format and content.

1.69 Practice Resources

Before attempting these questions, we highly recommend practicing on:

- **Killercoda CKA Scenarios** ⭐ Free hands-on practice environments
- **killer.sh CKA Simulator** - Included with exam registration

1.70 Instructions

- The CKA exam is **performance-based** (hands-on), not multiple choice
- Practice these scenarios in a real Kubernetes cluster

- Time yourself - aim for efficiency
- Use imperative commands when possible to save time

1.71 Section 1: Cluster Architecture, Installation & Configuration (25%)

1.71.1 Question 1.1 - Cluster Upgrade

Upgrade the control plane node from Kubernetes 1.29.0 to 1.30.0.

Show Solution

```
# Upgrade kubeadm
sudo apt-mark unhold kubeadm
sudo apt-get update && sudo apt-get install -y kubeadm=1.30.0-1.1
sudo apt-mark hold kubeadm

# Plan and apply upgrade
sudo kubeadm upgrade plan
sudo kubeadm upgrade apply v1.30.0

# Upgrade kubelet and kubectl
sudo apt-mark unhold kubelet kubectl
sudo apt-get update && sudo apt-get install -y kubelet=1.30.0-1.1
        kubectl=1.30.0-1.1
sudo apt-mark hold kubelet kubectl

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

1.71.2 Question 1.2 - etcd Backup

Create a backup of etcd to `/backup/etcd-snapshot.db`.

Show Solution

```
ETCDCTL_API=3 etcdctl snapshot save /backup/etcd-snapshot.db \
  --endpoints=https://127.0.0.1:2379 \
  --cacert=/etc/kubernetes/pki/etcd/ca.crt \
  --cert=/etc/kubernetes/pki/etcd/server.crt \
  --key=/etc/kubernetes/pki/etcd/server.key

# Verify backup
ETCDCTL_API=3 etcdctl snapshot status /backup/etcd-snapshot.db --
        write-out=table
```

1.71.3 Question 1.3 - RBAC

Create a Role named `pod-reader` in the `development` namespace that allows `get`, `list`, and `watch` on pods. Then create a RoleBinding to bind this role to user `jane`.

Show Solution

```
# Create namespace if not exists
kubectl create namespace development

# Create role
kubectl create role pod-reader \
  --verb=get,list,watch \
  --resource=pods \
  -n development

# Create rolebinding
kubectl create rolebinding read-pods \
  --role=pod-reader \
  --user=jane \
  -n development

# Verify
kubectl auth can-i list pods -n development --as jane
```

1.71.4 Question 1.4 - Join Worker Node

A new worker node needs to join the cluster. Generate the join command.

Show Solution

```
# On control plane
kubeadm token create --print-join-command

# Output will be something like:
# kubeadm join <control-plane-ip>:6443 --token <token> --
        discovery-token-ca-cert-hash sha256:<hash>

# Run the output command on the worker node with sudo
```

1.72 Section 2: Workloads & Scheduling (15%)

1.72.1 Question 2.1 - Node Affinity

Create a deployment named `web-app` with 3 replicas using image `nginx:1.21`. The pods should only be scheduled on nodes with label `disk=ssd`.

Show Solution

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: web-app
  template:
    metadata:
      labels:
        app: web-app
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: disk
                operator: In
                values:
                - ssd
      containers:
      - name: nginx
        image: nginx:1.21

# Or label a node first
kubectl label nodes <node-name> disk=ssd
```

1.72.2 Question 2.2 - Taints and Tolerations

Taint node `node01` with `key=value:NoSchedule`. Then create a pod named `tolerant-pod` that can be scheduled on this node.

Show Solution

```yaml
# Taint the node
kubectl taint nodes node01 key=value:NoSchedule

apiVersion: v1
kind: Pod
metadata:
  name: tolerant-pod
spec:
  tolerations:
  - key: "key"
    operator: "Equal"
    value: "value"
    effect: "NoSchedule"
  containers:
  - name: nginx
    image: nginx
```

1.72.3 Question 2.3 - DaemonSet

Create a DaemonSet named `log-collector` using image `fluentd:v1.14` that runs on all nodes including control plane nodes.

Show Solution

```yaml
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: log-collector
spec:
  selector:
    matchLabels:
      name: log-collector
  template:
    metadata:
      labels:
        name: log-collector
    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: fluentd
        image: fluentd:v1.14
```

1.72.4 Question 2.4 - Static Pod

Create a static pod named `static-nginx` using image `nginx` on node `node01`.

Show Solution

```bash
# SSH to node01
ssh node01

# Find static pod path
cat /var/lib/kubelet/config.yaml | grep staticPodPath
# Usually: /etc/kubernetes/manifests

# Create static pod manifest
cat <<EOF > /etc/kubernetes/manifests/static-nginx.yaml
apiVersion: v1
kind: Pod
metadata:
  name: static-nginx
spec:
  containers:
  - name: nginx
```

```
    image: nginx
EOF
```

1.73 Section 3: Services & Networking (20%)

1.73.1 Question 3.1 - Create Service

Create a deployment named `web` with image `nginx:1.21` and 3 replicas. Expose it using a NodePort service on port 30080.

Show Solution

```
# Create deployment
kubectl create deployment web --image=nginx:1.21 --replicas=3

# Expose as NodePort
kubectl expose deployment web --port=80 --type=NodePort --
        name=web-service

# Or with specific nodePort:
kubectl create service nodeport web-service --tcp=80:80 --node-
        port=30080
# Then patch selector if needed
```

1.73.2 Question 3.2 - Network Policy

Create a NetworkPolicy named `deny-all` in namespace `secure` that denies all ingress traffic to pods in that namespace.

Show Solution

```
kubectl create namespace secure
```

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

1.73.3 Question 3.3 - Ingress

Create an Ingress named `app-ingress` that routes: - `app.example.com/api` to service `api-svc` port 80 - `app.example.com/web` to service `web-svc` port 80

Show Solution

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: app-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  ingressClassName: nginx
  rules:
  - host: app.example.com
    http:
      paths:
      - path: /api
        pathType: Prefix
        backend:
          service:
            name: api-svc
            port:
              number: 80
      - path: /web
        pathType: Prefix
        backend:
          service:
            name: web-svc
            port:
              number: 80
```

1.73.4 Question 3.4 - CoreDNS

A pod cannot resolve service DNS names. Troubleshoot and fix the issue.

Show Solution

```bash
# Check CoreDNS pods
kubectl get pods -n kube-system -l k8s-app=kube-dns

# Check CoreDNS logs
kubectl logs -n kube-system -l k8s-app=kube-dns

# Check CoreDNS service
kubectl get svc -n kube-system kube-dns

# Check CoreDNS ConfigMap
kubectl get configmap coredns -n kube-system -o yaml

# Test DNS from a pod
kubectl run test --image=busybox:1.36 --rm -it -- nslookup
        kubernetes

# If CoreDNS pods are not running, check events
kubectl describe pods -n kube-system -l k8s-app=kube-dns
```

```
# Restart CoreDNS if needed
kubectl rollout restart deployment coredns -n kube-system
```

1.74 Section 4: Storage (10%)

1.74.1 Question 4.1 - PersistentVolume and PVC

Create a PersistentVolume named `pv-data` with 1Gi storage using hostPath `/data`. Then create a PVC named `pvc-data` that requests 500Mi.

Show Solution

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-data
spec:
  capacity:
    storage: 1Gi
  accessModes:
  - ReadWriteOnce
  hostPath:
    path: /data
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-data
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 500Mi
```

1.74.2 Question 4.2 - Pod with PVC

Create a pod named `data-pod` using image `nginx` that mounts the PVC `pvc-data` at `/usr/share/nginx/html`.

Show Solution

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

```
    volumeMounts:
    - name: data
      mountPath: /usr/share/nginx/html
  volumes:
  - name: data
    persistentVolumeClaim:
      claimName: pvc-data
```

1.75 Section 5: Troubleshooting (30%)

1.75.1 Question 5.1 - Pod Troubleshooting

A pod named `broken-pod` in namespace `default` is not running. Identify and fix the issue.

Show Solution

```
# Check pod status
kubectl get pod broken-pod

# Check events and details
kubectl describe pod broken-pod

# Common issues to look for:
# - ImagePullBackOff: Check image name, pull secrets
# - CrashLoopBackOff: Check logs
# - Pending: Check node resources, taints, affinity

# Check logs
kubectl logs broken-pod
kubectl logs broken-pod --previous

# If image issue, fix the image
kubectl set image pod/broken-pod <container>=<correct-image>

# Or edit the pod
kubectl edit pod broken-pod
```

1.75.2 Question 5.2 - Node Troubleshooting

Node `node01` is in NotReady state. Troubleshoot and fix.

Show Solution

```
# Check node status
kubectl describe node node01

# SSH to the node
ssh node01
```

```
# Check kubelet status
sudo systemctl status kubelet

# Check kubelet logs
sudo journalctl -u kubelet -f

# Common fixes:
# Start kubelet if stopped
sudo systemctl start kubelet
sudo systemctl enable kubelet

# Check container runtime
sudo systemctl status containerd
sudo systemctl start containerd

# Check disk space
df -h

# Check memory
free -m
```

1.75.3 Question 5.3 - Service Troubleshooting

A service named `web-svc` is not routing traffic to pods. Troubleshoot.

Show Solution

```
# Check service
kubectl get svc web-svc
kubectl describe svc web-svc

# Check endpoints
kubectl get endpoints web-svc

# If no endpoints, check:
# 1. Pod labels match service selector
kubectl get pods --show-labels
kubectl get svc web-svc -o yaml | grep selector -A 5

# 2. Pods are running
kubectl get pods -l <selector>

# 3. Pod ports match targetPort
kubectl get pods -o yaml | grep containerPort

# Fix selector if needed
kubectl patch svc web-svc -p '{"spec":{"selector":{"app":"correct-
        label"}}}'
```

1.75.4 Question 5.4 - Control Plane Troubleshooting

The API server is not responding. Troubleshoot.

Show Solution

```
# Check if API server pod is running
sudo crictl ps | grep kube-apiserver

# Check static pod manifest
sudo cat /etc/kubernetes/manifests/kube-apiserver.yaml

# Check API server logs
sudo crictl logs <container-id>

# Or check kubelet logs for static pod issues
sudo journalctl -u kubelet | grep apiserver

# Common issues:
# - Certificate expired: kubeadm certs renew all
# - Wrong configuration in manifest
# - etcd not accessible

# Check etcd
sudo crictl ps | grep etcd
ETCDCTL_API=3 etcdctl endpoint health \
  --endpoints=https://127.0.0.1:2379 \
  --cacert=/etc/kubernetes/pki/etcd/ca.crt \
  --cert=/etc/kubernetes/pki/etcd/server.crt \
  --key=/etc/kubernetes/pki/etcd/server.key
```

1.75.5 Question 5.5 - Application Logs

View the logs of pod `app-pod` container `sidecar` from the last hour.

Show Solution

```
kubectl logs app-pod -c sidecar --since=1h
```

1.76 Exam Tips

1. **Use aliases**: `alias k=kubectl`
2. **Enable auto-completion**: `source <(kubectl completion bash)`
3. **Use `--dry-run=client -o yaml`** to generate YAML templates
4. **Bookmark important docs** before the exam
5. **Practice on Killercoda** for free hands-on scenarios
6. **Focus on troubleshooting** - it's 30% of the exam
7. **Know etcd backup/restore** commands
8. **Practice cluster upgrades** with kubeadm

1.77 Additional Practice

- Killercoda CKA Scenarios - Free interactive scenarios
- killer.sh - Exam simulator (included with registration)
- Kubernetes Documentation - Allowed during exam

← Back to CKA Overview
