Certified Kubernetes Administrator (CKA) Certification Study Notes

Code: CKA

Cluster Architecture (25%)

Kubernetes Architecture

Core components of a Kubernetes cluster

Control Plane Components

Component	Purpose	Key Facts
kube-apiserver	REST API frontend for all cluster operations	All cluster communication goes through API server
etcd	Distributed key-value store for cluster state	Backup etcd regularly - it's the source of truth
kube-scheduler	Assigns pods to nodes based on constraints	Considers resources, affinity, taints/tolerations
kube-controller-manager	Runs controller processes (node, replication)	Ensures desired state matches actual state

Node Components

- **kubelet:** Ensures containers run in pods, communicates with API se

- **kube-proxy:** Maintains network rules for service connectivity
- **Container Runtime:** Runs containers (containerd, CRI-O)

> **Exam Tip**
>
> Know the location of static pod manifests: `/etc/kubernetes/manifests/`
>
> Control plane components run as static pods on kubeadm clusters.

kubeadm Cluster Setup

Initialize and manage Kubernetes clusters with kubeadm

Initialize Control Plane

> **Common Commands**

```
# Initialize control plane
kubeadm init --pod-network-cidr=10.244.0.0/16

# Set up kubectl for current user
mkdir -p $HOME/.kube
cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

# Join worker nodes
kubeadm join <control-plane>:6443 --token <token> \
  --discovery-token-ca-cert-hash sha256:<hash>

# Generate new join token
kubeadm token create --print-join-command
```

Cluster Upgrades

- **Step 1:** Upgrade kubeadm on control plane
- **Step 2:** Run kubeadm upgrade plan and apply
- **Step 3:** Drain node, upgrade kubelet/kubectl
- **Step 4:** Restart kubelet, uncordon node

- **Repeat:** For each node in the cluster

etcd Backup & Restore

Critical backup and restore procedures for etcd

> **High-Weight Topic**
>
> etcd backup/restore is heavily tested on the CKA exam!

Backup etcd

```
ETCDCTL_API=3 etcdctl snapshot save backup.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.db
```

Restore etcd

```
# Restore to new data directory
ETCDCTL_API=3 etcdctl snapshot restore backup.db \
  --data-dir=/var/lib/etcd-restored

# Update etcd manifest to use new data-dir
# Edit /etc/kubernetes/manifests/etcd.yaml
```

RBAC Configuration

Role-Based Access Control for Kubernetes

RBAC Objects

Role

Defines permissions within a namespace

ClusterRole

Defines permissions cluster-wide

RoleBinding

Binds Role to users/groups in a namespace

ClusterRoleBinding

Binds ClusterRole to users/groups cluster-wide

Example Role & RoleBinding

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

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

# Check permissions
kubectl auth can-i create pods --as=jane
kubectl auth can-i --list --as=system:serviceaccount:default:mysa
```

Workloads & Scheduling (15%)

Deployments

Managing stateless applications

Deployment Operations

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

# Scale
kubectl scale deployment nginx --replicas=5

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

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

Update Strategies

Strategy	Description	Use Case
RollingUpdate	Gradually replaces pods	Zero-downtime updates (default)
Recreate	Terminates all pods before creating new ones	When you can't run multiple versions

ConfigMaps & Secrets

Managing configuration and sensitive data

Creating ConfigMaps & Secrets

```
# ConfigMap
kubectl create configmap app-config \
  --from-literal=KEY1=value1 \
  --from-file=config.txt

# Secrets
kubectl create secret generic db-secret \
  --from-literal=password=mypassword
kubectl create secret tls tls-secret --cert=tls.crt --key=tls.key
```

Using in Pods

- **envFrom:** Import all keys as environment variables
- **env.valueFrom:** Import specific keys as env vars
- **volumeMounts:** Mount as files in container

Pod Scheduling

Control where pods run in the cluster

Scheduling Methods

nodeSelector

Simple label-based node selection

nodeAffinity

Advanced node selection with operators

podAffinity/Anti-Affinity

Schedule based on other pod locations

Taints & Tolerations

Repel pods from nodes unless tolerated

Taints & Tolerations

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

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

# Taint Effects: NoSchedule, PreferNoSchedule, NoExecute
```

Services & Networking (20%)

Service Types

Exposing applications in Kubernetes

Service Types

Type	Description	Access
ClusterIP	Internal cluster IP (default)	Within cluster only
NodePort	Exposes on each node's IP at static port	NodeIP:NodePort (30000-32767)
LoadBalancer	Cloud provider load balancer	External IP from cloud
ExternalName	Maps to external DNS name	CNAME record

Create Services

```
# Expose deployment
kubectl expose deployment nginx --port=80 --target-port=8080 --ty

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

Ingress

HTTP/HTTPS routing to services

Ingress Components

- **Ingress Controller:** Must be installed (nginx, traefik, etc.)
- **Ingress Resource:** Defines routing rules
- **IngressClass:** Specifies which controller to use

Key Features

Path-based routing

Route /api to api-service, /web to web-service

Host-based routing

api.example.com vs web.example.com

TLS termination

Configure HTTPS with secrets

Network Policies

Control traffic flow between pods

> **Important**
>
> Network Policies require a CNI plugin that supports them (Calico, Cilium, etc.)
>
> Flannel does NOT support Network Policies.

Default Behaviors

- **No policy:** All traffic allowed (deny nothing)
- **Policy with podSelector:** Only matching rules allowed
- **Empty podSelector :** Applies to all pods in namespace

Policy Types

Type	Controls
Ingress	Incoming traffic to selected pods
Egress	Outgoing traffic from selected pods

CoreDNS

DNS resolution in Kubernetes

DNS Names

Resource	DNS Format
Service	<service>.<namespace>.svc.cluster.local
Pod	<pod-ip-dashed>.<namespace>.pod.cluster.local

Test DNS Resolution

```
kubectl run test --image=busybox -it --rm -- nslookup my-service
kubectl run test --image=busybox -it --rm -- nslookup my-service.
```

Storage (10%)

PersistentVolumes & Claims

Persistent storage in Kubernetes

Storage Workflow

- **1. Admin creates PV:** Cluster-level storage resource
- **2. User creates PVC:** Request for storage
- **3. PVC binds to PV:** Based on capacity and access modes
- **4. Pod uses PVC:** Mount as volume in pod

Access Modes

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

Reclaim Policies

Retain

PV preserved after PVC deleted (manual cleanup)

Delete

PV and underlying storage deleted with PVC

Recycle (deprecated)

Basic rm -rf (use dynamic provisioning instead)

StorageClass

Dynamic storage provisioning

Dynamic Provisioning

- **StorageClass:** Defines provisioner and parameters
- **PVC with storageClassName:** Triggers dynamic PV creation
- **No manual PV needed:** Provisioner creates PV automatically

Volume Binding Modes

Mode	Description
Immediate	PV created immediately when PVC created
WaitForFirstConsumer	PV created when pod using PVC is scheduled

Troubleshooting (30%)

Cluster Component Debugging

Diagnose control plane and node issues

> **Highest Weight Domain**
>
> Troubleshooting is 30% of the exam - practice extensively!

Check Cluster Health

```
# Cluster status
kubectl get nodes
kubectl cluster-info
kubectl get componentstatuses  # Deprecated but may appear

# Control plane logs (kubeadm)
kubectl logs -n kube-system kube-apiserver-<node>
kubectl logs -n kube-system kube-scheduler-<node>
kubectl logs -n kube-system kube-controller-manager-<node>
kubectl logs -n kube-system etcd-<node>

# Node components
journalctl -u kubelet -f
journalctl -u containerd -f
```

Application/Pod Debugging

Diagnose pod and application issues

Pod Status Commands

```
# Get pod info
kubectl get pods -o wide
kubectl describe pod <pod-name>
kubectl logs <pod-name> -c <container> --previous
kubectl logs <pod-name> -f

# Interactive debugging
kubectl exec -it <pod-name> -- /bin/sh
kubectl exec -it <pod-name> -c <container> -- /bin/bash
```

```
# Port forwarding
kubectl port-forward pod/<pod-name> 8080:80
```

Common Pod Issues

Status	Common Causes	Debug Steps
Pending	Insufficient resources, node selector, PVC not bound	describe pod, check events
CrashLoopBackOff	Application error, config issues	kubectl logs --previous
ImagePullBackOff	Wrong image, registry auth	Check image name, imagePullSecrets
CreateContainerConfigError	Missing ConfigMap/Secret	Check referenced ConfigMaps/Secrets exist

Node Troubleshooting

Diagnose and manage node issues

Node Management

```
# Node status
kubectl describe node <node-name>
kubectl get node <node-name> -o yaml

# Drain and cordon
kubectl cordon <node-name>       # Mark unschedulable
kubectl drain <node-name> --ignore-daemonsets --force
kubectl uncordon <node-name>     # Mark schedulable
```

Node Conditions

Ready

Node is healthy and can accept pods

MemoryPressure

Node memory is low

DiskPressure

Node disk capacity is low

PIDPressure

Too many processes on node

kubectl Tips & Tricks

Essential commands for the exam

Set Up Aliases (Exam)

```
# Add to ~/.bashrc
alias k=kubectl
alias kgp='kubectl get pods'
alias kgs='kubectl get svc'
alias kgn='kubectl get nodes'
export do='--dry-run=client -o yaml'

# vim settings (~/.vimrc)
set expandtab
set tabstop=2
set shiftwidth=2
```

Generate YAML Quickly

```
# Generate pod YAML
kubectl run nginx --image=nginx --dry-run=client -o yaml > pod.ya

# Generate deployment YAML
kubectl create deployment nginx --image=nginx --dry-run=client -o

# Context switching
kubectl config get-contexts
kubectl config use-context <context>
kubectl config set-context --current --namespace=<ns>
```

> **Documentation Access**
>
> You can access **kubernetes.io/docs** during the exam!
>
> Bookmark the kubectl cheatsheet: kubernetes.io/docs/reference/kubectl/cheatsheet/