

KUBERNETES STUDY GUIDE 2026



Shruthi Chikkela
Learnwithshruthi

KA Study Guide: Kubernetes Certified Administrator (2026)

This guide is designed for learning and understanding Kubernetes fundamentals and real-world cluster administration. It covers cluster setup, workloads, networking, storage, security, and troubleshooting, with an emphasis on practical usage rather than certification objectives. The content includes explanations, examples, and commonly used commands to support effective self-study. No practice tasks are included.

Introduction

Overview

The CKA exam tests your ability to administer Kubernetes clusters, focusing on architecture, installation, workloads, scheduling, networking, storage, and troubleshooting. This guide provides clear, practical content to master these skills using `kubectl`, YAML, kubeadm, Helm, and Customize.

- Use `k` shorthand for `kubectl` to save time (e.g., `k get pods`).
- Always specify `-n <namespace>` for namespaced commands to avoid errors (e.g., `k -n kube-system get pods`).
- Practice `kubectl edit` for quick YAML modifications.
- Time management is key; skip tough questions and revisit.

Core Concepts

etcd

Overview

- **What is etcd?** Distributed key-value store for cluster data (e.g., pod states), running on control plane (ports 2379, 2380).
- **Why Important?** Stores cluster state; critical for backups.

Key Concepts

- **API:** v3 (current, e.g., `etcdctl put`); v2 deprecated.
 - **TLS:** Requires `--cacert`, `--cert`, `--key`.
 - **Commands:** `etcdctl get`, `put`, `snapshot save/restore`.
 - Memorize TLS flags and endpoint (`127.0.0.1:2379`), Practice snapshot commands.
-

Pods

Overview

- **What is a Pod?** Smallest unit, hosting one or more containers with shared network/storage.
- **Why Important?** Core workload component.

Key Concepts

- **Commands:** k run, k get pods, k describe pod.
- **InitContainers:** Run setup tasks before main containers.
- **Sidecars:** Co-located containers for logging, monitoring.

Example: Create Pod

Scenario: Deploy nginx

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  namespace: default
spec:
  containers:
    - name: nginx
```

pod.

Apply: k apply -f nginx-pod.yaml -n default

Example: InitContainer

Scenario: Add init container to setup pod.

```
apiVersion: v1
kind: Pod
metadata:
  name: init-pod
  namespace: default
spec:
  initContainers:
    - name: init-
      setup image:
        busybox
      command: ["/bin/sh", "-c", "echo 'Setup complete' > /data/init.txt"]
      volumeMounts:
        - name: data
          mountPath: /
```



```
image: nginx
volumeMounts
:
- name: data
  mountPath: /
  data
```

- Use `k edit` for quick pod updates.
 - Always include `-n <namespace>` (e.g., `k get pods -n default`).
 - Sidecars are common for logging (e.g., streaming logs for `kubectl logs`).
 - InitContainers run once, ideal for setup tasks.
-

ReplicaSets

Overview

- **What are ReplicaSets?** Ensure specified pod replicas, replacing Replication Controllers.
- **Why Important?** Manage scaling and availability.

Key Concepts

- **Selector:** `matchLabels` for pod selection.
- **Commands:** `k scale`, `k`

get rs. Example: Scale

ReplicaSet Scenario:

```
k scale rs new-replica-set -n default --replicas=5
```

Scale to 5 replicas.

- Check selectors with `k describe rs -n <namespace>`.
-

Deployments

Overview

- **What are Deployments?** Manage stateless apps with updates and rollbacks.
- **Why Important?** Standard for scalable workloads.

Key Concepts

- **Strategy:** RollingUpdate (default), Recreate.
- **Commands:** k create deployment, k set image, k rollout.

Example: Update

Deployment Scenario:

```
k set image deployment/nginx nginx=nginx:1.20 -n default  
k rollout status deployment/nginx -n default
```

Update nginx image.

Example: Add Sidecar to Deployment

Scenario: Update synergy-deployment with busybox sidecar for logging.

```
k edit deployment synergy-deployment -n default
```

Add to spec.template.spec:

```
volumes:  
- name: shared-  
  logs emptyDir:  
  {}  
containers:  
- name: sidecar  
  image: busybox:stable  
  command: ["/bin/sh", "-c", "tail -n+1 -f /var/log/synergy-deployment.log"]  
volumeMounts:
```

Note: Ensure main container mounts shared-logs at /var/log.

Services

Overview

- **What are Services?** Provide stable IPs for pods (ClusterIP, NodePort, LoadBalancer, ExternalName).
- **Why Important?** Enable communication.

Key Concepts

- **Types:** ClusterIP (internal), NodePort (external), LoadBalancer (cloud), ExternalName (DNS).
- **Commands:** k expose, k get svc.

Example: Create ClusterIP

Scenario: Expose nginx

```
k expose deployment nginx -n default --port=80 --target-port=80  
deployment.
```

Example: Expose Deployment with NodePort

Scenario: Expose nginx deployment on NodePort.

```
k expose deployment nginx -n default --port=80 --target-port=80 --type=NodePort
```

Output: Service nginx with port 80 and node port (30000–32767). Verify:

```
k get svc -n default
```

- Verify endpoints: k get ep -n <namespace>.
-

ConfigMaps and Secrets

Overview

- What are They?** ConfigMaps store configs; Secrets store sensitive data.
- Why Important?** Decouple configuration.

Key Concepts

- ConfigMap:** Key-value pairs or files, mounted as volumes/env.
- Immutable ConfigMaps:** Prevent updates post-creation, common in exam.
- Secrets:** Base64-encoded.
- Commands:** k create configmap/
secret.

Example: Create

Immutable ConfigMap Scenario:

```
apiVersion: v1  
kind: ConfigMap  
metadata:  
  name: app-config  
  namespace: default  
  immutable: true  
data:
```

Store immutable app settings.

```
Apply: k apply -f configmap.yaml -n default
```

Example: Edit Immutable ConfigMap

Scenario: Update immutable ConfigMap (requires deletion/recreation).

```
k delete configmap app-config -n default
```

Recreate with updated data:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: app-config
  namespace: default
immutable: true
data:
```

- **Immutable ConfigMaps can't be edited; delete and recreate.**
 - Use `k create configmap --from-literal` for quick creation.
 - Check immutability: `k describe configmap -n <namespace>`.
-

Scheduling

Taints and Tolerations

Overview

- **What are They?** Taints restrict pods; tolerations allow scheduling on tainted nodes.
- **Why Important?** Control placement.

Key Concepts

- **Effects:** `NoSchedule`, `PreferNoSchedule`, `NoExecute`.
- **Commands:** `k taint`, `k describe node`.

Example: Taint Node

Scenario: Block pods on

```
k taint nodes node01 key1=value1:NoSchedule
node01.
```

- Check taints: `kubectl describe node`.

Node Selectors and Affinity

Overview

- **What are They?** Node selectors assign pods to labeled nodes; affinity offers advanced rules.
- **Why Important?** Optimize placement.

Key Concepts

- **Node Selector:** Key-value match (e.g., `disk=ssd`).
- **Affinity:** Node/pod affinity/anti-affinity.

Example: Node Selector

Scenario: Schedule on SSD

```
apiVersion: v1
kind: Pod
metadata:
  name: ssd-pod
  namespace: default
spec:
  nodeSelector:
    disk: ssd
  containers:
    - name: nginx
```

nodes.

- Use `kubectl label node` to add labels.

Priority Classes

Overview

- **What are Priority Classes?** Assign priority to pods for scheduling/preemption.
- **Why Important?** Prioritize critical workloads; frequent in CKA 2025.

Key Concepts

- **Fields:** value (priority), preemptionPolicy (PreemptLowerPriority/Never), globalDefault, description.
- **Default:** Priority 0 unless globalDefault set.
- **System Classes:** system-cluster-critical (2000000000), system-node-critical (2000001000).
- **Commands:** k create priorityclass, k patch.

Example: Create PriorityClass

Scenario: Create high-priority (value 100000).

```
k create priorityclass high-priority --value=100000 --description="For XYZ pods"
```

Example: Patch Deployment

Scenario: Assign high-priority to nginx deployment.

```
k patch deployment nginx -n default --patch
'{"spec": {"template": {"spec": {"priorityClassName": "high-priority"}}}}'
```

Verify:

```
k get pods -n default -o custom-
columns="NAME:.metadata.name,PRIORITY:.spec.priorityClassName"
```

- Use `k create priorityclass` for speed.
- Patch is faster than `k edit` for simple updates.
- Common in CKA 2025 for resource contention scenarios.

Logging & Monitoring

Logging

Overview

- **What is Logging?** Captures container logs for debugging.
- **Why Important?** Diagnose app issues.

Key Concepts

- **Commands:** k logs, k logs -f.
- **Tools:** Fluentd, Loki, Elasticsearch.

Example: View Logs

Scenario: Check nginx logs.

```
k logs nginx -n default
```

Example: Multi-Container Logs

Scenario: Check sidecar logs in synergy-deployment.

```
k logs synergy-deployment-<pod> -n default --container=sidecar
```

- Use `--container` for sidecars (e.g., logging containers).
 - Helpful for verifying sidecar functionality.
-

Monitoring

Overview

- **What is Monitoring?** Tracks metrics (CPU, memory).
- **Why Important?** Detects performance issues.

Key Concepts

- **Tools:** Metrics Server, Prometheus, Grafana.
- **Commands:** `k top node/`

`pod`. **Example: Node**

Metrics Scenario: Check

```
k top node  
CPU/memory.
```

Application Lifecycle

Jobs and CronJobs

Overview

- **What are They?** Jobs run tasks to completion; CronJobs schedule tasks.
- **Why Important?** Manage batch workloads.

Key Concepts

- **Job:** Ensures completion.

- **CronJob:** Cron syntax (e.g., */5 * * * *).
- **Commands:** `k create job/cronjob`.

Example: Create Job

Scenario: Run

```
apiVersion: batch/v1
kind: Job
metadata:
  name:
    pi
  namespace: default
spec:
  template
  :
  spec:
    containers:
      - name: pi
```

computation.

- Use `restartPolicy: Never` for Jobs.
-

DaemonSets

Overview

- **What are DaemonSets?** Run one pod per node (e.g., logging, monitoring).
- **Why Important?** Cluster-wide services.

Key Concepts

- **Use Cases:** Fluentd, Prometheus node exporter.
- **Commands:** `k create daemonset`

`daemonset`. Example: Create

DaemonSet Scenario:

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: fluentd
  namespace: default
```

Deploy Fluentd.


```
selector:
  matchLabels:
    app: fluentd
template:
  metadata
  :
  labels
  :
    app: fluentd
spec:
```

- Verify: `k get pods -o wide -n <namespace>`.
-

StatefulSets

Overview

- **What are StatefulSets?** Manage stateful apps with stable identities/storage.
- **Why Important?** Databases, queues.

Key Concepts

- **Features:** Stable names (e.g., `mysql-0`), headless services.
- **Commands:** `k create statefulset`

Example: Create

StatefulSet Scenario: Deploy

MySQL.

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: mysql
  namespace: default
spec:
  serviceName: mysql
  replicas: 3
  selector:
    matchLabels
      : app:
        mysql
```

```
labels:  
  app: mysql  
spec:  
  containers:  
  - name: mysql  
    image: mysql
```

Horizontal Pod Autoscaler (HPA)

Overview

- **What is HPA?** Automatically scales pods based on metrics (e.g., CPU).
- **Why Important?** Optimizes resource usage; common in CKA 2025.

Key Concepts

- **API Versions:**
 - v1: Basic CPU scaling, no behavior or custom metrics.
 - v2: Supports custom metrics, behavior (e.g., stabilization window).
- **Fields:** minReplicas, maxReplicas, targetCPUUtilizationPercentage (v1), metrics, behavior (v2).
- **Commands:** kubectl create hpa, kubectl autoscale.

Example: Create HPA

Scenario: Scale apache-server in autoscale namespace for 50% CPU, min=1, max=4, 30s downscale stabilization.

```
apiVersion: autoscaling/v2  
kind:  
  HorizontalPodAutoscaler  
metadata:  
  name: apache-  
    server  
  namespace:  
    autoscale  
spec:  
  scaleTargetRef:  
    apiVersion: apps/v1  
    kind: Deployment  
    name: apache-server  
  minReplicas: 1  
  maxReplicas: 4  
  metrics:  
  - type: Resource  
    resource:
```

```
    averageUtilization:  
      50 behavior:  
        scaleDown:  
          stabilizationWindowSeconds: 30
```

Apply: `k apply -f hpa.yaml -n autoscale`

- Use `v2` for metrics/behavior; `v1` fails for advanced configs.
 - Verify: `k get hpa -n <namespace>`.
 - Common in CKA 2025 for scaling scenarios.
-

Cluster Maintenance

Cluster Upgrades

Overview

- **What is Cluster Maintenance?** Upgrading Kubernetes, managing nodes, backups.
- **Why Important?** Ensures stability/security.

Key Concepts

- **Process:** Upgrade control plane, then nodes.
- **Commands:** `kubeadm upgrade plan/apply`.
- **Strategy:** Upgrade to next stable version (e.g., 1.31 to 1.32).

Example: Check Upgrade

Plan Scenario: Plan upgrade

```
kubeadm upgrade plan  
to v1.32.
```

- Drain nodes: `k drain <node> --ignore-daemonsets`.
- Always target next stable version.

Node Management

Key Concepts

- **Commands:** `k cordon/uncordon`, `k drain`.
- **Drain:** Evicts pods safely.

Example: Drain Node

```
k drain node01 --ignore-daemonsets
```

Scenario: Prepare

node01.

- Use `--ignore-daemonsets` for DaemonSets.
-

Security

RBAC

Overview

- **What is RBAC?** Role-Based Access Control for resource security.
- **Why Important?** Restricts permissions.

Key Concepts

- **Resources:** Role/ClusterRole, RoleBinding/ClusterRoleBinding.
- **Commands:** `kubectl create role/rolebinding`.

Example: Create Role

Scenario: Allow pod get in default.

```
apiVersion:  
rbac.authorization.k8s.io/v1 kind:  
Role  
metadata:  
  namespace: default  
  name: pod-reader  
rules:  
- apiGroups: [ "" ]
```

- **Test:** `kubectl auth can-i -n <namespace>`.
-

Network Policies

Overview

- **What are Network Policies?** Control pod traffic (ingress/egress).
- **Why Important?** Secure communication.

Key Concepts

- **Selectors:** Pod/namespace labels.
- **Types:** Ingress, Egress.

Example: Allow Ingress

Scenario: Allow traffic to

```
apiVersion: networking.k8s.io/
v1 kind: NetworkPolicy
metadata:
  name: allow-web
  namespace: default
spec:
  podSelector:
    matchLabels:
      app: web
  policyTypes:
  - Ingress
  ingress:
    app=web.
```

- : Flannel does not support network policies; use Calico/Weave.

Storage

Persistent Volumes and Claims

Overview

- **What are PVs/PVCs?** PVs provide storage; PVCs request storage.
- **Why Important?** Persistent data for apps.

Key Concepts

- **PV:** Cluster-wide storage.

- **PVC:** Namespace-scoped request.
- **Commands:** k get pvc/

pvc. **Example: Create**

PVC Scenario:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc
  namespace: default
spec:
  accessModes:
    -
      ReadWriteOnce
  resources:
```

Request 1Gi.

- Check binding: k describe pvc -n <namespace>.
-

Storage Classes

Overview

- **What are Storage Classes?** Define storage types for dynamic PVs.
- **Why Important?** Automate

storage. **Example: Create**

StorageClass Scenario:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name:
    ssd
provisioner: kubernetes.io/aws-ebs
```

Define SSD storage.

Key Points to Remember for the CKA Exam

Only One Default StorageClass Allowed

If another StorageClass is already default, you must remove its default status first before setting a

new one.

Check existing defaults with: `kubectl get storageclass`

Look for (default) in the output.

Required Annotation

The exact annotation must be used:

metadata:

 annotations:

 storageclass.kubernetes.io/is-default-class: "true"

Two Ways to Set Default

Imperative (Quick for Exam):

```
kubectl patch storageclass <name> -p '{"metadata": {"annotations":{"storageclass.kubernetes.io/is-default-class":"true"}}}'
```

Declarative (YAML for Version Control):

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

```
kind: StorageClass
```

```
metadata:
```

```
  name: fast
```

```
  annotations:
```

 storageclass.kubernetes.io/is-default-class: "true"

```
provisioner: kubernetes.io/aws-ebs
```

```
parameters:
```

```
  type: gp3
```

Removing Default Status

If you need to remove the default status (e.g., to set a new default):

```
kubectl patch storageclass <current-default> -p '{"metadata": {"annotations":{"storageclass.kubernetes.io/is-default-class":"false"}}}'
```

Custom Resource Definitions (CRDs)

Overview

- **What are CRDs?** Extend Kubernetes API with custom resources.
- **Why Important?** Used for apps like cert-manager; common in CKA 2025.

Key Concepts

- **Structure:** Define custom kinds (e.g., certificate).
- **Commands:** `k get crd`, `k explain`.

Example: Verify cert-manager CRDs

Scenario: List CRDs and extract certificate subject field.

```
k get crd -o yaml > ~/resources.yaml  
k explain Certificate.spec.subject > ~/subject.yaml
```

resources.yaml:


```
certificates.cert-manager.io
challenges.acme.cert-
```

subject.yaml:

```
Documentation for Certificate.spec.subject
```

- Use `k explain` for CRD details.
- Common for cert-manager scenarios in exam.

Resources and Quotas

Overview

- **What are Resources and Quotas?** Define pod resource requests/limits and namespace quotas.
- **Why Important?** Manage resource allocation.

Key Concepts

- **Requests/Limits:** CPU, memory for pods.
- **ResourceQuota:** Limits namespace resources.
- **Commands:** `k describe node`, `k create quota`.

Example: Allocate Resources for WordPress

Scenario: Divide resources (CPU: 1, memory: 2015360Ki) across 3 WordPress pods in `relative-fawn`, with overhead.

1. Calculate per pod: CPU: 0.3 (90% of 1 / 3), Memory: 604608Ki (90% of 2015360 / 3).
2. Scale to 0:

```
3. k scale deployment wordpress -n relative-fawn --replicas=0
```

4. Edit deployment:

```
5. k edit deployment wordpress -n relative-fawn
```

Update `spec.template.spec`:

```
containers:
- name:
  wordpress
  resources:
```

```
cpu: "300m"
memory:
"604608Ki"
initContainers:
- name: init-
wordpress
resources:
```

6. Scale back:

```
7. k scale deployment wordpress -n relative-fawn --replicas=3
```

8. Verify:

```
9. k get pods -n relative-fawn -o wide
```

- Scale to 0 for safe updates.
- Ensure requests match for containers/initContainers.

Networking

Linux Networking Basics

Overview

- **What is Linux Networking?** Manages interfaces, IPs, routing.
- **Why Important?** Underpins Kubernetes networking; bridge networking common in CKA 2025.

Key Concepts

- **Components:** Switch (L2), Router (L3), Gateway.
- **Commands:** ip link, ip addr, ip route.

Example: Enable Bridge

Networking Scenario: Configure

bridge (exam).

```
cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
br_netfilter
EOF
cat <<EOF | sudo tee /etc/sysctl.d/
k8s.conf
net.bridge.bridge-nf-call-
ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
EOF
```

```
sudo sysctl --system
```

Example: List

Interfaces Scenario:

```
ip link
```

Check interfaces.

Output:

```
1: lo: <LOOPBACK,UP> mtu 65536  
3: eth0@if9157: <BROADCAST,UP> mtu 1410
```

DNS in Linux and Kubernetes

Overview

- **What is DNS?** Resolves names to IPs.
- **Why Important?** Service discovery.

Key Concepts

- **Linux:** /etc/hosts, /etc/resolv.conf.
- **Kubernetes:** CoreDNS resolves <service>. <namespace>. svc.cluster.local.
- **Commands:** nslookup,

dig. Example: Test

DNS Scenario:

```
k exec busybox -n default -- nslookup nginx.default.svc.cluster.local  
Resolve nginx.
```

Output:

```
Server: 172.20.0.10  
Name: nginx.default.svc.cluster.local  
Address: 172.20.100.123
```

- Ensure port 53 (TCP/UDP) open.

Network Namespaces

Overview

- **What are Network Namespaces?** Isolate network configs.
- **Why Important?** Pod networking foundation.

Key Concepts

- **Creation:** ip netns add.
- **Connection:** veth pairs.
- **Commands:** ip netns list/

exec. Example: Connect

Namespaces Scenario:

```
ip netns add ns1
ip netns add ns2
ip link add veth-ns1 type veth peer name veth-
ns2 ip link set veth-ns1 netns ns1
ip link set veth-ns2 netns ns2
ip netns exec ns1 ip addr add 192.168.1.1/24 dev
veth-ns1 ip netns exec ns2 ip addr add
192.168.1.2/24 dev veth-ns2 ip netns exec ns1 ip
link set veth-ns1 up
```

Link ns1 and ns2.

Docker Networking

Overview

- **What is Docker Networking?** Manages container communication.
- **Why Important?** Kubernetes builds on it.

Key Concepts

- **Modes:** Bridge (docker0), Host, None.
- **Commands:** docker network ls/inspect.

Example: Bridge

Network Scenario:

```
docker run -d --name web1 nginx
docker network inspect bridge
```

Inspect bridge.

- Bridge uses NAT.
- Container Networking Interface (CNI)

Overview

- **What is CNI?** Standard for pod networking.
- **Why Important?** Pod-to-pod communication.

Key Concepts

- **Plugins:** Bridge, IPAM, Flannel, Calico.
- **Config:** /etc/cni/

net.d/. **Example:**

Check Flannel

```
cat /etc/cni/net.d/10-flannel.conflist
```

Scenario: Inspect

config.

- First config file used.

Cluster Node

Networking Overview

- **What is Node Networking?** Node/pod connectivity.
- **Why Important?** Cluster communication.

Key Concepts

- **Ports:**
 - Control Plane: 6443 (API), 2379–2380 (etcd), 10250 (kubelet).
 - Worker: 10250, 30000–32767 (NodePort).
 - CNI: Flannel (8472/UDP), Weave (6783/TCP, 6784/UDP).
- **Plugins:** Flannel, Weave,

Calico. Example: Install

Weave Scenario: Deploy

```
kubectl apply -f https://github.com/weaveworks/weave/releases/download/v2.8.1/weave-daemonset-
```

Weave Net.

- Memorize ports.

Pod Networking

Overview

- **What is Pod Networking?** Assigns IPs/interfaces to pods.
- **Why Important?** Pod communication.

Key Concepts

- **Setup:** Unique IP per pod, shared namespace.
- **Inter-Pod:** CNI bridge/

overlay. **Example:**

Inspect Pod IP

```
k get pod nginx -n default -o wide
k exec nginx -n default -- ip addr
```

Scenario: Check nginx IP.

- Verify IPs: `k get pod -o wide -n <namespace>`.

IP Address Management (IPAM)

Overview

- **What is IPAM?** Allocates pod IPs.
- **Why Important?** Prevents conflicts/exhaustion.

Key Concepts

- **Types:** host-local, dhcp, static.
- **Issues:** IP exhaustion.

Example: Check

```
k get ds weave-net -n kube-system -o yaml
```

IPAM Scenario:

Inspect Weave.

- Verify PodCIDR in CNI.

Services Networking (kube-proxy)

Overview

- **What is kube-proxy?** Manages service IPs/load balancing.
- **Why Important?** Service access.

Key Concepts

- **Modes:** iptables, IPVS.
- **Config:** /var/lib/kube-proxy/config.conf.

Example: Create

Service Scenario:

```
k expose deployment nginx -n default --port=80 --target-port=80  
Expose nginx.
```

- Check endpoints: k get ep -n <namespace>.

Cluster DNS

Overview

- **What is Cluster DNS?** Resolves service/pod names via CoreDNS.
- **Why Important?** Service discovery.

Key Concepts

- **CoreDNS:** Runs in kube-system, port 53.
- **Records:** <service>.<namespace>.svc.cluster.local.

Example: Test DNS

Scenario: Resolve nginx.

```
k exec busybox -n default -- nslookup nginx.default.svc.cluster.local
```

- Verify CoreDNS: k get pods -n kube-system.
- _____

Ingress and Gateway API

Overview

- **What are They?** Ingress for HTTP/HTTPS; Gateway API for HTTP, TCP, UDP.
- **Why Important?** External access.

Key Concepts

- **Ingress:** Uses controller (e.g., NGINX), annotations.
- **Gateway API:** `GatewayClass`, `Gateway`, `HTTPRoute`.
- **TLS:** Common in Gateway API exam scenarios.

Example: Create Ingress

Scenario: Route `/wear` to `wear-service`.

```
kubectl create ingress ingress-test -n app-space \
--rule="wear.my-online-store.com/wear*=wear-service:80" \
```

Example: Ingress for echoserver-service

Scenario: Expose `echoserver-service` in `echo-sound` on `http://example.org/echo`.

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: echo
  namespace: echo-sound
spec:
  rules:
  - host:
    example.org
    http:
      paths:
      - path: /echo
        pathType: Prefix
        backend:
          service:
            name: echoserver-service
```

Apply: `kubectl apply -f ingress.yaml -n echo-sound` **Verify:**

```
curl -o /dev/null -s -w "%{http_code}\n" http://example.org/echo
```

Output: 200

Example: Gateway with TLS

Scenario: Expose frontend-svc with TLS using certificateRefs.

```
apiVersion: gateway.networking.k8s.io/
v1 kind: Gateway
metadata:
  name: nginx-gateway
  namespace: nginx-
  gateway
spec:
  gatewayClassName: nginx
  listeners:
    - name: https
      protocol: HTTPS
      port: 443
      tls:
        mode: Terminate
        certificateRefs:
          - kind: Secret
            name: frontend-tls
            namespace: nginx-
            gateway
  allowedRoutes:
    namespaces:
      from: All
---
apiVersion: gateway.networking.k8s.io/
v1 kind: HTTPRoute
metadata:
  name: frontend-route
  namespace: nginx-
  gateway
spec:
  parentRefs:
    - name: nginx-gateway
      namespace: nginx-
      gateway
  rules:
    - matches:
        - path:
            type: PathPrefix
            value: /
  backendRefs:
    - name: frontend-
      svc port: 80
```

Apply: k apply -f gateway.yaml -n nginx-gateway

- Troubleshoot: `kubectl describe ingress -n <namespace>`.
- **TLS with `certificateRefs` is common in CKA 2025.**

Design and Install HA Cluster

Designing a Kubernetes Cluster

Overview

- **What is Cluster Design?** Planning HA cluster with redundancy.
- **Why Important?** Ensures reliability.

Key Concepts

- **HA Setup:** 3+ control plane nodes, load balancer (6443), external/stacked etcd.
- **Networking:** PodCIDR (e.g., 172.17.0.0/16), ServiceCIDR (e.g., 172.20.0.0/16), CNI.
- **Node Specs:** 2 CPU, 4GB RAM for control plane.
- Avoid CIDR overlaps.

Install Kubernetes

Deploying with kubeadm

Overview

- **What is kubeadm?** Bootstraps Kubernetes clusters.
- **Why Important?** Standard CKA setup.

Key Concepts

- **Steps:** Configure kernel, install tools, initialize, join nodes, deploy CNI.
- **Options:** --pod-network-cidr, --service-cidr, --apiserver-advertise-address.

Example: Configure Nodes

Scenario: Enable bridge

```
cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
br_netfilter
EOF
cat <<EOF | sudo tee /etc/sysctl.d/
k8s.conf net.bridge.bridge-nf-call-
ip6tables = 1
```

networking.

```
sudo sysctl --system
```

Example: Install Tools

Scenario: Install v1.32.

```
sudo apt-get update
sudo apt-get install -y apt-transport-https ca-certificates curl
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.32/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:/v1.32/deb/ /' | sudo tee
/etc/apt/sources.list.d/
kubernetes.list sudo apt-get update
sudo apt-get install -y kubelet=1.32.0-1.1 kubeadm=1.32.0-1.1 kubectl=1.32.0-1.1
```

Example: Initialize Cluster

Scenario: Bootstrap on 192.168.233.162.

```
kubeadm init --apiserver-advertise-address 192.168.233.162 --apiserver-cert-extra-
sans=controlplane \
--pod-network-cidr 172.17.0.0/16 --service-cidr
172.20.0.0/16 mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
```

Example: Install

Flannel Scenario:

Deploy Flannel.

```
curl -LO https://raw.githubusercontent.com/flannel-io/flannel/v0.20.2/
Documentation/kube-
```

Update net-conf.json:

```
net-conf.json: |
{
  "Network": "172.17.0.0/16",
  "Backend": {
    "Type": "vxlan"
  }
}
```

Add args:

```
args:
|- --ip-masq
```

```
- --kube-subnet-mgr  
- --iface=eth0
```

Apply: k apply -f kube-flannel.yml

Example: Join Worker

Scenario: Join node01.

```
kubeadm join 192.168.233.162:6443 --token  
j7hnsg.oz15uyij7z2ov32w.svc.cluster.local \
```

Example: Verify

```
k get nodes
```

- Match PodCIDR with CNI.
- Save `kubeadm join` token.

Helm Basics (2025)

Overview

- **What is Helm?** Package manager for Kubernetes apps.
- **Why Important?** Simplifies deployments with charts.
- **Benefits:** Templates YAML, supports versioning, repos.

Key Concepts

- **Components:** Helm CLI, Charts, Releases, Repositories.
- **Chart:** `Chart.yaml` (metadata), `values.yaml`, `templates/`.
- **Commands:** `helm install`, `upgrade`, `rollback`, `repo`.
- **Helm 3:** Client-side, no Tiller, 2-way merge.

Example: Install Helm

Scenario: Install Helm.

```
curl https://baltocdn.com/helm/signing.asc | gpg --dearmor > /usr/share/keyrings/  
helm.gpg  
sudo apt-get install apt-transport-https --yes  
echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/  
helm.gpg] https://baltocdn.com/helm/stable/debian/ all main" | sudo tee
```

```
sudo apt-get update  
sudo apt-get install helm
```

Example: Install Argo CD

Scenario: Install Argo CD (v7.3) in argocd, no CRDs.

Add repo:

```
helm repo add argo https://argoproj.github.io/argo-helm
```

Generate template:

```
helm template argocd argo/argo-cd --version 7.3.3 --namespace  
argocd --set  
crds.install=false > /argo-helm.yaml
```

Install:

```
helm install argocd argo/argo-cd --version 7.3.3 -n argocd --create-  
namespace --set  
crds.install=false
```

Example: Add Bitnami

```
helm repo add bitnami https://charts.bitnami.com/bitnami
```

Example: Search ArtifactHub

Scenario: Find Consul.

```
helm search hub consul | grep hashicorp
```

Example: Install Apache

```
helm install amaze-surf bitnami/apache -n default
```

Example: List Releases

```
helm list -n default
```

Example: Uninstall

```
helm uninstall happy-browse -n default
```

Example: Upgrade/Rollback

```
helm upgrade dazzling-web bitnami/nginx --version 18.3.6 -n default  
helm rollback dazzling-web 3 -n default
```

- Use --dry-run to preview.
 - Common for Argo CD installation in CKA 2025.
-

Kustomize Basics

Overview

- **What is Kustomize?** Customizes manifests with patches/overlays.
- **Why Important?** Manages multi-env deployments.
- **Kustomize vs. Helm:** Patches vs. templates; Kustomize for custom apps.

Key Concepts

- **Kustomization:** kustomization.yaml defines resources/patches.
- **Base/Overlays:** Base for shared, overlays for specific envs.
- **Commands:** kustomize build, k apply -k.
- **Transforms:** commonLabels, namePrefix, images.
- **Patches:** Strategic Merge, JSON6902.

Example: Install Kustomize

```
curl -s "https://raw.githubusercontent.com/kubernetes-sigs/kustomize/master/hack/install_kustomize.sh" | bash
```

Example: Multi-Folder

Scenario: Manage nginx, db, message-broker.

```
# kustomization.yaml
resources:
- db
- message-broker

# db/kustomization.yaml
resources:
- db-config.yaml
- db-depl.yaml
- db-service.yaml
```

Example: Transformers

```
commonLabels:
```

```
app: my-app
namePrefix: prod-
namespace:
production images:
- name: nginx
  newTag: 1.21
```

Example: Strategic Merge Patch

```
# api-patch.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: api-deployment
spec:
  template
  :
  spec:
    containers:
```

Example: JSON Patch

```
resources:
- mongo-
  depl.yaml
patches:
- target:
  kind: Deployment
  name: mongo-
  deployment patch: |-
```

Example: Base/Overlays

```
# base/kustomization.yaml
resources:
- api-deployment.yaml
- db-configMap.yaml

# overlays/qa/kustomization.yaml
resources:
- ../../base
commonLabel
ls:
environment:
qa patches:
- target:
  kind: Deployment
```

```
- op: replace
  path: /spec/template/spec/containers/0/image
  value: caddy
```

Example: Apply

```
k apply -k k8s/overlays/staging/
```

- Preview: kustomize build.
-

Troubleshooting

Common Issues

Overview

- **What is Troubleshooting?** Diagnosing cluster, pod, network issues.
- **Why Important?** Maintains cluster health.

Key Concepts

- **Tools:** `k describe`, `k logs`, `crictl`.
- **Issues:** Pending, CrashLoopBackOff, DNS

failures. **Example: Troubleshoot kube-**

scheduler Scenario: CrashLoopBackOff.

```
k describe pod kube-scheduler-controlplane -n kube-system
```

Fix: Edit `/etc/kubernetes/manifests/kube-scheduler.yaml`.

Example: Troubleshoot

kubelet Scenario:

`kubelet.service` fails.

```
systemctl status
kubelet
journalctl -u kubelet -n 100
```

Fix: Check `/var/lib/kubelet/config.yaml`, CNI, CRI.

Example: Use crictl

Scenario: `kubectl` fails; check containers.

```
cricctl      # Running  
cricctl ps -a # All
```

Example: Troubleshoot

CoreDNS Scenario:

CrashLoopBackOff.

```
k get pods -n kube-system -l k8s-app=kube-dns
```

Fix SELinux:

```
k -n kube-system get deployment coredns -o yaml | sed 's/allowPrivilegeEscalation:  
false/allowPrivilegeEscalation: true/g' | k apply -f -
```

Fix DNS:

```
k edit configmap coredns -n kube-  
system # Replace: forward . /etc/
```

Example: troubleshoot kube-proxy

Scenario: Service connectivity.

```
k get pods -n kube-system -l k8s-app=kube-  
proxy k logs -n kube-system kube-proxy-
```

- Use `cricctl ps -a` when `kubectl` fails.
- Start with `k describe`, `k logs`.

Network Plugins

Concepts

- **Plugins:**
 - Flannel: Overlay, no policies.
 - Weave: Overlay, policies.
 - Calico: Policies, routing.
- **Install:**
 - Flannel: `k apply -f https://raw.githubusercontent.com/coreos/flannel/.../kuben-flannel.yml`
 - Weave: `k apply -f https://github.com/weaveworks/weave/.../weave-daemonset.yaml`
 - Calico: `k apply -f https://docs.projectcalico.org/.../calico.yaml`

- Check `/etc/cni/net.d/`.

DNS Troubleshooting

Concepts

- **CoreDNS**: Service account, deployment, ConfigMap (`corefile`).
- **Port**: 53.

Example: Check Endpoint

```
k get ep kube-dns -n kube-system
```

- Ensure CNI for CoreDNS.

kube-proxy Troubleshooting

Concepts

- **kubeProxy**: DaemonSet, iptables/IPVS.
- **Config**: `/var/lib/kubeadm/config.conf`.
- : Check ConfigMap

Other Topics

JSON Path

Overview

- **What is JSON Path?** Queries JSON/YAML in `kubectl` outputs.
- **Why Important?** Filters output.

Key Concepts

- **Syntax**: Dot (`.metadata.name`), bracket (`[]`).
- **Command**: `k get -o jsonpath="{<>}"`.

Example: List Pods

```
k get pods -n default -o jsonpath=".items[*].metadata.name"
```

Example: Filter by Label

```
k get pods -n default -o  
jsonpath=".items[?(@.metadata.labels.app=='web')].metadata.name"
```

Example: Node CPU

```
k get nodes -o jsonpath=".items[*].status.allocatable.cpu"
```

Cluster Administration

Overview

- **What is Cluster Administration?** Manages backups, certificates.
- **Why Important?** Cluster integrity.

Example: Backup etcd

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

- **Practice backups.**

FOR ANY REALTIME HANDSON PROJECTS

+ Follow



Shruthi Chikkela
Learnwithshruthi