How to Pass the CKA 2025 Exam - Questions & Tips

1. Introduction

Hey everyone! Today, we're diving into real exam questions with tips to help you crush it like a Kubernetes ninja! Let's roll!

2. What's the Exam Like?

The **CKA** is 100% hands-on. You'll work on Kubernetes clusters, running `kubectl` commands and editing YAML files. It's 2 hours long with about 16 questions.

The questions cover:

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

Pro Tip: Use `k` instead of `kubectl` to save time. Example:

```
k get pods -n default
```

3. How to Prepare?

To ace the exam, practice like a pro. I used the **KodeKloud** CKA course, and their labs are just like the real exam!

https://learn.kodekloud.com/user/courses/cka-certification-course-certified-kubernetes-administrator

https://learn.kodekloud.com/user/courses/ultimate-certified-kubernetes-administrator-cka-mock-exam-series

Their mock exams show you where you stand. Other resources:

- **Killer code**

  https://killercoda.com/cka

- **Killer.sh** for exam simulations

Pro Tip: Practice `k edit` and `k apply` daily. Speed is key! 💪

4. Exam Questions & Tips

Let’s check out questions like the ones in the exam, with tips to solve them fast. We’ll use commands and YAML you’ll see in the real deal.

1. Setting Up kubeadm and cri-dockerd

Task: Prepare the system for Kubernetes by installing cri-dockerd and configuring system parameters.

Steps:

```
sudo dpkg -i ~/cri-dockerd_0.3.9.3-0.ubuntu-jammy_amd64.deb
sudo systemctl enable cri-docker
sudo systemctl start cri-docker
sudo sysctl -w net.bridge.bridge-nf-call-iptables=1
sudo sysctl -w net.ipv6.conf.all.forwarding=1
sudo sysctl -w net.ipv4.ip_forward=1
sudo sysctl -w net.netfilter.nf_conntrack_max=131072
```

```
# sysctl params required by setup, params persist across reboots
cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.ipv4.ip_forward = 1
net.bridge.bridge-nf-call-iptables=1
net.ipv6.conf.all.forwarding=1
net.netfilter.nf_conntrack_max=131072

EOF

# Apply sysctl params without reboot
sudo sysctl --system
```

Tip: Verify services are running:

```
systemctl status cri-docker
```

https://kubernetes.io/docs/setup/production-environment/container-runtimes/

2. Ingress for echoserver-service

Task: Create an Ingress named `echo` in namespace `echo-sound` for service `echoserver-service` at `http://example.org/echo`.

Steps:

```
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
            port:
              number: 8080
```

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

Tip: Double-check namespace and service name. If curl fails:

```
k describe ingress echo -n echo-sound
```

https://kubernetes.io/docs/concepts/services-networking/ingress/

---

3. Installing CNI (Calico)

Task**: I**nstall Calico (v3.29.2) as CNI with Network Policy support.

Steps:

```
k create -f https://raw.githubusercontent.com/projectcalico/calico/v3.29.2/manifests/tigera-
operator.yaml
k get pods -n tigera-operator
```

Tip: Calico is ideal for Network Policies. Ensure pods are running.

---

4. Argo CD with Helm

Task: Generate a Helm template for Argo CD (v7.7.3) in namespace argocd without CRDs.

Steps:

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

helm repo update
helm repo list
helm search repo argo

```
helm template argocd argo/argo-cd --version 7.7.3 --namespace argocd --set crds.install=false >
/argo-helm.yaml
```

---

5. PriorityClass for busybox-logger

Task: Create a PriorityClass named `high-priority` with a value one less than the highest user-defined priority class, and apply it to `busybox-logger`.

Steps:

```
k get priorityclass
```

If the highest value is 1000000000:

```
apiVersion: scheduling.k8s.io/v1
kind: PriorityClass
metadata:
  name: high-priority
value: 999999999
globalDefault: false
```

```
k apply -f priorityclass.yaml
k edit deployment busybox-logger -n priority
```

Add:

```
spec:
  template:
    spec:
      priorityClassName: high-priority
```

```
k rollout restart deployment busybox-logger -n priority
```

Tip: Pod evictions are normal with PriorityClass.

https://kubernetes.io/docs/concepts/scheduling-eviction/pod-priority-preemption/

---

6. Service and NodePort for front-end

Task: Modify the `front-end` Deployment in namespace `sp-culator` to expose port 80/tcp, and create a Service named `front-end-svc`.

Steps:

```
k -n sp-culator edit deployment front-end
```

Edit `spec.template.spec.containers[0].ports` to add:

```
ports:
- containerPort: 80
  name: http
  protocol: TCP
```

Create Service:

```
apiVersion: v1
kind: Service
metadata:
  name: front-end-svc
  namespace: sp-culator
spec:
  selector:
    app: front-end
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80
    nodePort: 30080
  type: NodePort
```

```
k apply -f front-end-svc.yaml
k get svc -n sp-culator
```

Tip: Ensure the Service selector matches Deployment labels.

https://kubernetes.io/docs/concepts/services-networking/service/#type-nodeport

---

7. StorageClass (low-latency)

Task: Create a StorageClass named `low-latency` with provisioner `rancher.io/local-path` and set it as default.

Steps:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: low-latency
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
provisioner: rancher.io/local-path
volumeBindingMode: WaitForFirstConsumer
```

```
k apply -f low-latency-sc.yaml
k get sc
```

Tip: Delete any old default StorageClass:

```
k delete sc <name>
```

https://kubernetes.io/docs/concepts/storage/storage-classes/

---

8. Sidecar for synergy-deployment

Task: Add a sidecar to `synergy-deployment` using `busybox:stable`.

Steps:

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

Add:

```
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:
        - name: shared-logs
          mountPath: /var/log
```

```
k logs <pod-name> -c sidecar -n default
```

https://kubernetes.io/docs/concepts/workloads/pods/sidecar-containers/

---

9. cert-manager CRDs

Task: Verify cert-manager and extract `subject` for Certificate CRD.

Steps:

```
k get pods -n cert-manager
```

```
k get crd | grep cert-manager > ~/resources.yaml
k explain certificate.spec.subject > ~/subject.yaml
```

Tip: Use `k explain` if CRDs confuse you.

---

10. WordPress Resource Requests

Task: Adjust WordPress in `relative-fawn` for 3 replicas

Steps:

1. Check allocatable resources:

```
kubectl describe node node01 | grep -A 5 Allocatable
```

Example: CPU: 2000m, Memory: 4000Mi.

cpu: 4
memory: 24000Mi

2. Calculate resources:
   - 10% for node: CPU = 200m, Memory = 400Mi
   - 90% for WordPress: CPU = 1800m, Memory = 3600Mi
   - Per pod (3 replicas): CPU = 600m, Memory = 1200Mi
3. Edit deployment:

```
kubectl edit deployment wordpress -n relative-fawn
```

Update:

```
spec:
  replicas: 3
  template:
    spec:
      containers:
      - name: wordpress
        resources:
          requests:
            cpu: 600m
            memory: 1200Mi
      initContainers:
      - name: init-wordpress
        resources:
          requests:
            cpu: 600m
            memory: 1200Mi
```

4. Verify:

```
kubectl get deployment wordpress -n relative-fawn
kubectl get pods -n relative-fawn
```

Tip: If pods fail to schedule, check node capacity:

```
kubectl describe node node01
```

11. MariaDB and PVC

Task: Recreate MariaDB in `mariadb` with a PVC.

Steps:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mariadb
  namespace: mariadb
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 250Mi
```

```
k apply -f mariadb-pvc.yaml
```

Edit `~/mariadb-deploy.yaml`:

```
spec:
  template:
    spec:
      volumes:
      - name: mariadb-storage
        persistentVolumeClaim:
          claimName: mariadb
      containers:
      - name: mariadb
        volumeMounts:
        - name: mariadb-storage
          mountPath: /var/lib/mysql
```

```
k apply -f mariadb-deploy.yaml
k get pods -n mariadb
```

Tip: Ensure the PV is available before creating the PVC.

https://kubernetes.io/docs/concepts/storage/persistent-volumes/

12. Migration from Ingress to Gateway API

Task: Migrate a web app from Ingress `web` to Gateway API, maintaining HTTPS, using GatewayClass `nginx`.

Steps:

1. Create Gateway `web-gateway`:

```yaml
apiVersion: gateway.networking.k8s.io/v1
kind: Gateway
metadata:
  name: web-gateway
  namespace: default
spec:
  gatewayClassName: nginx
  listeners:
  - name: https
    protocol: HTTPS
    port: 443
    hostname: gateway.web.k8s.local
    tls:
      mode: Terminate
      certificateRefs:
      - kind: Secret
        name: web-tls
```

2. Create HTTPRoute `web-route`:

```yaml
apiVersion: gateway.networking.k8s.io/v1
kind: HTTPRoute
metadata:
  name: web-route
  namespace: default
spec:
  parentRefs:
  - name: web-gateway
  hostnames:
  - gateway.web.k8s.local
  rules:
  - backendRefs:
    - kind: Service
      name: web
      port: 80
```

3. Apply:

```
k apply -f web-gateway.yaml
k apply -f web-route.yaml
```

4. Test:

```
curl https://gateway.web.k8s.local
```

5. Delete old Ingress:

```
k delete ingress web -n default
```

https://gateway-api.sigs.k8s.io/guides/tls/

13. HorizontalPodAutoscaler (HPA) for apache-server

Task: Create an HPA named `apache-server` in namespace `autoscale`, targeting 50% CPU, with 1-4 pods and a 30-second downscale stabilization window.

Steps:

```
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:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 50
  behavior:
    scaleDown:
      stabilizationWindowSeconds: 30
```

```
k apply -f apache-hpa.yaml
k get hpa -n autoscale
```

Tip: Use `autoscaling/v2`. Verify the Deployment:

```
k get deployment apache-server -n autoscale
```

https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/

14. NGINX ConfigMap for TLSv1.3

Task: Update ConfigMap `nginx-config` in namespace `nginx-static` to allow only TLSv1.3 connections.

Steps:

apply both

```
k edit configmap nginx-config -n nginx-static
```

Modify `nginx.conf` to:

```
data:
  nginx.conf: |
    events {}
    http {
      server {
        listen 443 ssl;
        ssl_certificate /etc/nginx/tls/tls.crt;
        ssl_certificate_key /etc/nginx/tls/tls.key;
        ssl_protocols TLSv1.3; # TLSv1.2 removed
        location / {
          root /usr/share/nginx/html;
          index index.html;
        }
      }
    }
```

ssl_protocols TLSv1.2 TLSv1.3;

Restart deployment:

```
k rollout restart deployment nginx-static -n nginx-static
```

Test TLSv1.2 failure:

```
curl --tls-max 1.2 https://web.k8s.local
```

https://kubernetes.io/docs/concepts/configuration/configmap/

5. Exam Day Tips

On exam day, stay calm and organized:

- Review commands like `k edit`, `k apply`.
- Skip tough questions and return later.

- Always use `-n <namespace>`:

```
k get pods -n default
```

- If the cluster has issues:

```
crictl ps -a
```