CERTLABZ CERTIFICATION BOARD

Certified Container Orchestration Professional (CCOP)

Ten Sample Questions

These ten questions are written at the difficulty of the live exam. Six carry a figure, a data table, a configuration or a code listing; four are plain scenario questions, as on the live paper. Every one is a scenario a working platform engineer would meet, and none of them appears in the live question bank.

The live exam is multistage adaptive: how you score on one stage selects the difficulty of the next, so no two candidates sit the same paper. It is a performance-based exam: each form can carry up to 5 to 12 performance-based questions alongside the multiple-choice questions, which means you build, sequence or complete something rather than only choosing from a list. Answers and full explanations follow each question here; the live exam does not disclose item-level answers.

Question 1

Domain 3: Workloads, Scheduling and Configuration

CONFIG

Emma Wilson reviews a Deployment container spec.

```
containers:
- name: web
  image: web:latest          # mutable tag
  # no resources set
  # no readiness or liveness probe
```

Listing 1.1 The container spec as written.

Which two changes most improve reliability and predictability?

A. Increase the replica count and add a NodePort service, so more copies run and the app is reachable from outside the cluster.

B. Pin the image to a specific version and add resource requests and readiness probes, so scheduling and rollouts behave predictably.

C. Move the container to a DaemonSet and mount a host path, so a copy runs on every node with fast local storage available to it.

D. Add more memory limits only and keep the latest tag, since limits alone are enough to make deployments fully predictable.

Correct answer: B
Pinning the image gives reproducible deployments instead of a moving latest tag, resource requests let the scheduler place the pod well, and readiness probes stop traffic reaching pods before they can serve and keep rolling updates safe. More replicas or a DaemonSet do not fix the missing versioning, sizing and health signals.

Question 2

Domain 2: Kubernetes Architecture and Core Objects

DATA

James Carter matches workloads to the right Kubernetes object.

Workload	Requirement
Stateless web app	Rolling updates, many replicas
Clustered database	Stable identity and per-pod storage
Log agent	One copy on every node
Nightly report	Run on a schedule

Table 2.1 Workloads and requirements.

Which mapping is correct?

A. Web app StatefulSet, database Deployment, log agent CronJob, report DaemonSet, matching each workload to a workload object.

B. Web app Deployment, database StatefulSet, log agent DaemonSet, report CronJob, matching each workload to the right object.

C. Web app DaemonSet, database CronJob, log agent Deployment, report StatefulSet, spreading the workloads across the objects.

D. Web app CronJob, database DaemonSet, log agent StatefulSet, report Deployment, assigning each workload a different object.

Correct answer: B
A stateless web app fits a Deployment, a clustered database needs a StatefulSet for stable identity and per-pod storage, a per-node log agent is a DaemonSet, and a scheduled report is a CronJob. The other options misassign these, for example putting a stateless web app on a StatefulSet or a database on a CronJob.

Question 3

Domain 4: Networking, Services and Ingress

CONFIG

Olivia Davis checks pod-to-pod connectivity assumptions in a new cluster.

```
cluster state:
  network policies : none applied
  CNI              : supports NetworkPolicy
question: can frontend pods reach the payments pods?
```

Listing 3.1 Cluster networking state.

What is true by default, and what should be done?

A All pod traffic is denied by default, so an explicit allow policy must be added before frontend can reach payments at all.

B All pod traffic is allowed by default, so frontend can reach payments; apply default-deny then allow only the required flows.

C Pods can only talk within their own node by default, so a LoadBalancer service is needed for frontend to reach payments.

D Traffic is encrypted and restricted by default, so no NetworkPolicy is needed and payments is already properly isolated.

Correct answer: B
Kubernetes networking is open by default: with no policies, any pod can reach any other, so frontend can currently reach payments. To segment sensitive workloads, apply a default-deny NetworkPolicy per namespace and then allow only the specific flows required. Kubernetes does not deny or encrypt pod traffic by default.

Question 4 Domain 5: Storage, Security and Access Control DATA

Liam Anderson designs storage for a database running in Kubernetes.

Option	Behaviour
Container filesystem	Ephemeral, lost when the pod is rescheduled
emptyDir volume	Tied to the pod lifetime
PVC on durable storage	Survives pod rescheduling
ConfigMap	Holds non-sensitive config, not data

Table 4.1 Storage options.

The database must keep its data when its pod is rescheduled to another node. Which option fits, and how is it usually delivered?

A The container filesystem, because data written inside the container is automatically preserved across pod rescheduling in Kubernetes.

B An emptyDir volume, because it provides fast local storage that persists independently of the pod that created it originally.

C A PersistentVolumeClaim on durable storage, typically via a StatefulSet, because it survives pod rescheduling to another node.

D A ConfigMap, because mounting configuration as a volume is the standard way to give a database durable data storage in a cluster.

Correct answer: C
Only a PersistentVolumeClaim bound to durable storage keeps data when a pod moves to another node, and it is usually delivered through a StatefulSet for stable identity. The container filesystem and emptyDir are both ephemeral, and a ConfigMap holds configuration, not the database's data.

Question 5 — Domain 5: Storage, Security and Access Control — CONFIG

Ava Thompson reviews a ServiceAccount used by a small web application pod.

```
serviceAccount: web-app
binding: web-app -> ClusterRole cluster-admin
# the pod only needs to read one ConfigMap
```

Listing 5.1 The pod's access binding.

What is wrong, and what is the fix?

A. Nothing is wrong, because binding cluster-admin to application pods is the standard way to avoid permission errors at runtime.

B. The binding grants far more than needed; if the pod is compromised it controls the cluster, so scope it to read that one ConfigMap.

C. The only problem is the ServiceAccount name, and renaming it to something more descriptive resolves the security concern here.

D. The pod needs a second cluster-admin binding for redundancy, so that access continues if the first binding is ever removed.

Correct answer: B
A pod that only needs to read one ConfigMap should not hold cluster-admin: if it is compromised, the attacker controls the whole cluster. The fix is least privilege, a Role granting read on that single ConfigMap bound to the ServiceAccount. The name is irrelevant and a second admin binding would make it worse.

Question 6 — Domain 6: Observability, Scaling and Operations — DATA

A service is autoscaled but does not keep up during traffic spikes.

Fact	Value
HPA metric	CPU utilisation
CPU during spikes	Low (20%)
Real bottleneck	Message queue backlog grows
Latency	Rises with backlog

Table 6.1 Autoscaling behaviour during spikes.

Why is autoscaling failing to respond, and what fixes it?

A. CPU is the wrong signal here; the bottleneck is the queue, so the HPA should scale on a custom metric such as queue length.

B. The HPA needs higher CPU limits, because raising the CPU limit will make the autoscaler add replicas during the traffic spikes.

C. Autoscaling should be disabled and replicas fixed at a high number, because the HPA cannot handle spiky workloads at all.

D The image is too large, so the pods start too slowly, and shrinking the image is what will let autoscaling respond in time.

Correct answer: A
The pods sit at 20% CPU while the queue backlog grows, so scaling on CPU never triggers even though the service is overloaded. The HPA can scale on custom or external metrics, so scaling on queue length, the true bottleneck, is the fix. Raising CPU limits, fixing replicas high or shrinking the image do not address the wrong scaling signal.

Question 7 Domain 1: Containers and Images

A container writes important data to its own local filesystem, then the pod is rescheduled to a different node. What happens to the data, and what should have been used?

A The data moves with the container automatically, so the local filesystem is a perfectly acceptable place to store important data.

B The data is lost because the container filesystem is ephemeral; durable data must be written to a persistent volume instead.

C The pod cannot be rescheduled while it holds local data, so the node keeps running the pod and the data stays safe on that node.

D The node retains the data forever and reattaches it, so no persistent volume is needed as long as the same node is available.

Correct answer: B
A container's filesystem is ephemeral and does not follow the container when it is rescheduled, so the data is lost. Durable data must be written to a persistent volume backed by real storage. Rescheduling is normal in Kubernetes and does not preserve local container data.

Question 8 Domain 3: Workloads, Scheduling and Configuration

A rolling update proceeds even though the new pods crash on startup, and the service goes down. Which misconfiguration best explains this?

A The image registry was unreachable, so Kubernetes could not pull the new image and therefore took the running service offline.

B Readiness probes were missing or wrong, so Kubernetes treated the crashing new pods as ready and shifted traffic to them.

C The namespace quota was exceeded, which caused the old healthy pods to be evicted before the new pods had started up.

D The nodes were tainted, so the scheduler could not place any pods at all and the entire service was left with no running copies.

Correct answer: B
Correct readiness probes tell Kubernetes when a new pod can actually serve, so a rollout waits for readiness before shifting traffic and terminating old pods. Without working readiness probes, crashing new pods are treated as ready and traffic moves to them, taking the service down. A registry, quota or taint problem would stop pods starting, not send traffic to broken ones.

Question 9

Domain 6: Observability, Scaling and Operations

An engineer needs to take a node down for maintenance without causing an outage. What is the correct sequence?

A. Delete the node immediately, because Kubernetes will always reschedule every affected pod instantly with no user impact at all.

B. Cordon the node, drain it so its pods reschedule gracefully while respecting disruption budgets, do the work, then uncordon it.

C. Stop the kubelet on the node without draining, because halting the agent is the fastest safe way to remove a node from service.

D. Scale every workload to a single replica first, then power off the node, so there is less traffic to disrupt during the maintenance.

Correct answer: B

The safe pattern is cordon (stop new pods scheduling), drain (evict existing pods gracefully, respecting pod disruption budgets so enough replicas stay up), perform the maintenance, then uncordon to return the node to service. Deleting or powering off a node without draining, or scaling down to one replica, risks an outage.

Question 10

Domain 5: Storage, Security and Access Control

A colleague says Secrets are safe because Kubernetes stores them base64-encoded in etcd. How should you respond?

A. Agree, because base64 encoding is a strong form of encryption that fully protects Secret values stored in the etcd datastore.

B. Explain that base64 is only encoding, not encryption, so enable encryption at rest for Secrets and restrict access with RBAC.

C. Agree, because Secrets never need protection beyond the defaults, as only cluster administrators can ever read them anyway.

D. Explain that Secrets should instead be stored in a ConfigMap, because ConfigMaps are the more secure place for sensitive values.

Correct answer: B

Base64 is an encoding that anyone can reverse; it provides no confidentiality. Protecting Secrets means enabling encryption at rest in etcd and restricting who can read them with RBAC. ConfigMaps are for non-sensitive data and are no more secure, so moving Secrets there would be worse.

Published by the CertLabz Certification Board, a division of CertLabz LLC. Companion to objectives document CCOP-001. https://certlabz.com/certifications/certified-container-orchestration-professional.html