Kubernetes Interview Questions PDF

Kubernetes interviews cover the architecture (control plane and nodes), pods, deployments, services, config maps and secrets, scaling, storage, ingress, rolling updates and troubleshooting with kubectl. This page gives short model answers with example commands and manifests, written for practice and based on stable, long-standing concepts.

Key takeaways

- You declare desired state in YAML and Kubernetes works to keep the cluster in that state.
- A pod is the smallest unit; deployments manage replicated pods and rolling updates.
- Services give stable access to changing pods; ingress routes external traffic.
- ConfigMaps and Secrets separate configuration from images.
- Troubleshooting starts with kubectl get, describe and logs.

What is Kubernetes interview questions?

Kubernetes is an open-source system for deploying, scaling and managing containerised applications across a cluster of machines by keeping the actual state of the system matched to a desired state you declare.

What is Kubernetes and what problem does it solve?

Running a few containers is easy; running hundreds across many machines needs scheduling, restarts, scaling, networking and updates. Kubernetes automates these. You describe what you want (for example, three copies of an app) and a controller keeps reality equal to that description, restarting or replacing pods as needed.

What is the architecture?

The **control plane** makes decisions: the API server (front door), etcd (stores cluster state), the scheduler (chooses nodes) and controller managers (reconcile state). **Nodes** run the workloads: the kubelet (talks to the control plane and runs pods), the container runtime and kube-proxy (handles service networking).

What is a pod?

A pod is the smallest deployable unit: one or more containers that share network and storage and run together on one node. Pods are temporary; if one dies, a controller creates a new one with a new IP address.

Deployment, ReplicaSet and StatefulSet?

A **Deployment** manages stateless pods through a ReplicaSet, handles rolling updates and rollbacks. A **ReplicaSet** keeps a set number of pod copies. A **StatefulSet** is for stateful apps that need stable names and storage, such as databases. A **DaemonSet** runs one pod per node, for agents like log collectors. A **Job** runs a task to completion and a **CronJob** runs jobs on a schedule.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web
spec:
  replicas: 3
  selector:
    matchLabels: { app: web }
  template:
    metadata:
      labels: { app: web }
    spec:
      containers:
      - name: web
        image: nginx:1.27
        ports: [{ containerPort: 80 }]
```

What are services?

Pod IPs change, so a **Service** gives a stable name and virtual IP that routes to pods matched by labels. Types: **ClusterIP** (inside the cluster, default), **NodePort** (opens a port on every node), **LoadBalancer** (uses a cloud load balancer) and **ExternalName**. An **Ingress** routes HTTP traffic by host or path, and needs an ingress controller.

How are configuration and secrets handled?

A **ConfigMap** holds non-sensitive settings, and a **Secret** holds sensitive values. Both can be injected as environment variables or mounted as files. Secrets are only base64 encoded by default, so enable encryption at rest, limit access with RBAC and consider an external secrets manager.

How does scaling work?

Manual: `kubectl scale deployment web --replicas=5`. Automatic: the **Horizontal Pod Autoscaler** changes replicas based on metrics such as CPU. The **Cluster Autoscaler** adds or removes nodes. Set resource **requests** (used for scheduling) and **limits** (caps) on containers so scaling and scheduling work well.

How do rolling updates and rollbacks work?

When you change the image in a Deployment, Kubernetes replaces pods gradually, keeping the app available (controlled by `maxSurge` and `maxUnavailable`). Check with `kubectl rollout status deployment/web` and undo with `kubectl rollout undo deployment/web`.

What are probes?

A **liveness** probe restarts a container that is stuck. A **readiness** probe controls whether a pod receives traffic. A **startup** probe gives slow apps time to start. Proper probes prevent traffic going to pods that are not ready.

How does storage work?

Containers' file systems are temporary. A **PersistentVolume** is storage in the cluster and a **PersistentVolumeClaim** is a request for it. A **StorageClass** provisions volumes dynamically. Choose access modes that match your app.

How do namespaces and RBAC help?

Namespaces split a cluster into virtual sections for teams or environments. RBAC defines who can do what through roles and bindings. Use least privilege.

How do I troubleshoot a failing pod?

```
kubectl get pods
kubectl describe pod <name>     # events, reasons
kubectl logs <name> --previous  # logs from the last crash
kubectl exec -it <name> -- sh
kubectl get events --sort-by=.lastTimestamp
```

Common states: **Pending** (no node fits or storage is missing), **ImagePullBackOff** (wrong image name or credentials), **CrashLoopBackOff** (the container keeps failing; read logs) and **OOMKilled** (memory limit exceeded).

What scenario questions come up?

An app works locally but pods keep restarting. Describe the pod, read previous logs, check probes and resource limits and compare configuration. **How would you do a zero-downtime release?** Use a rolling update with readiness probes, a pod disruption budget and a tested rollback. **How do you keep a database safe in the cluster?** Use a StatefulSet or a managed database, persistent volumes, backups and restricted network access.

How should I practise?

Run a small local cluster with a tool for local development, deploy a simple app, expose it with a service, scale it, break it on purpose and fix it using describe and logs. Practising failures builds the troubleshooting stories interviewers like.

Frequently asked questions

What is Kubernetes used for?

It automates deploying, scaling and managing containerised applications across a cluster by keeping actual state equal to the desired state.

What is a pod?

A pod is the smallest deployable unit in Kubernetes: one or more containers that share network and storage on a node.

What is the difference between a Deployment and a StatefulSet?

A Deployment manages stateless replicated pods, while a StatefulSet gives stable identities and storage to stateful apps such as databases.

Why do pods need a Service?

Pod IP addresses change, so a Service provides a stable name and address that routes to the matching pods.

What does CrashLoopBackOff mean?

A container keeps starting and failing. Check kubectl logs with the previous flag and describe the pod to find the cause.

Sources and further reading

- Kubernetes documentation
- Docker documentation

About this PDF

These PDFs are study material for learning and revision. They are not a certificate, a degree or a promise of a job, and technology changes, so check official documentation for the latest details. Spot an error? Write to info@uncodemy.com and the trainer team will review it.

More free resources and courses: uncodemy.com | info@uncodemy.com | +91 98183 66550