TECHNICAL INTERVIEW PLAYBOOK

Kubernetes Scenario-Based Interview Questions

The Senior SRE & Platform Engineer Field Handbook: The 4-Phase Diagnostic Heuristic, Technical STAR Structure, and 10 Production Battle Plans

Naveed Ahmed (Kumbhar)
Lead Cloud Platform Architect & Kubernetes Specialist
Official Publication · naveedkumbhar.com

Table of Contents

1. **The 4-Phase Diagnostic Heuristic Framework**
2. **The Technical STAR Framework for Kubernetes Scenarios**
3. **Scenario 1: Pod Stuck in Pending (Resource Quotas & Node Affinity)**
4. **Scenario 2: Rapid CrashLoopBackOff with Exit Code 137 (OOMKilled)**
5. **Scenario 3: Pod Running but Inaccessible (HTTP 503 Service Unavailable)**
6. **Scenario 4: Intermittent 5-Second Latency (CoreDNS conntrack race)**
7. **Scenario 5: Ingress 502 Bad Gateway During Rolling Deployments**
8. **Scenario 6: StatefulSet EBS Multi-Attach Volume Deadlocks**
9. **Scenario 7: AWS VPC CNI Secondary IP Exhaustion**
10. **Scenario 8: Mutating Webhook Timeout Freezing Cluster Deployments**
11. **Scenario 9: etcd Fsync Latency Spikes and Control Plane Stalls**
12. **Scenario 10: HPA Thrashing and Pod Flapping Under Spiky Traffic**
13. **The 60-Second Command Toolkit for Live Coding Screens**

1. The 4-Phase Diagnostic Heuristic Framework

Senior Kubernetes technical interviews do not test command memorization; they evaluate systematic reasoning under pressure. When an interviewer presents an ambiguous incident scenario, apply this 4-phase heuristic:

> **Phase 1: Triage & Blast Radius** — Identify whether the failure affects a single pod, a microservice tier, or the entire worker node pool.
> **Phase 2: Non-Destructive Telemetry** — Gather cluster events and historical logs without mutating state: `kubectl get events --sort-by=.metadata.creationTimestamp`.
> **Phase 3: Root Cause Isolation** — Correlate container exit codes (137 OOMKilled, 143 SIGTERM) with Linux cgroups and network overlays.
> **Phase 4: Remediation & Guardrails** — Propose immediate mitigation followed by automated architectural safeguards (LimitRanges, PDBs, WARM_IP_TARGET).

2. Common Failure Mode Triage Matrix

Pod Status / Error	Primary Diagnostic CLI Command	Root Cause Investigation	Architectural Prevention
CrashLoopBackOff	`kubectl logs <pod> --previous --tail=50`	Application panic, unhandled exception, or failing startup probe.	Tune startupProbe failureThreshold and add preStop hooks.
OOMKilled (Exit 137)	`dmesg -T | grep -i oom_reaper`	Process memory exceeded container limit; cgroup killer terminated pod.	Tune JVM heap / memory limits and set Kubelet system reservations.
CreateContainerConfigError	`kubectl describe pod <pod> | grep Error`	Referenced ConfigMap or Secret missing in namespace.	Validate CI/CD secret sync pipelines and ExternalSecrets.
Pending	`kubectl describe pod <pod> | grep -A 4 Events`	Insufficient CPU/memory, tainted nodes, or subnet IP exhaustion.	VPC CNI prefix delegation, Karpenter provisioners, and resource quotas.
502 Bad Gateway	`kubectl logs -n ingress-nginx <pod>`	SIGTERM sent before ingress proxy removes pod IP from upstreams.	preStop sleep buffer (15s) and graceful shutdown drain.

3. The Technical STAR Structure for Interviews

Structure your verbal answers using the technical STAR method to maintain clarity:

- **Situation:** Define the operational context and blast radius: *"During peak Black Friday traffic, 40% of checkout requests failed with HTTP 503."*
- **Task:** State your objective: *"My objective was to restore checkout availability within 5 minutes without risking data corruption."*
- **Action:** Walk through non-destructive telemetry, command executions, and hypothesis validation: *"I ran kubectl get endpoints to verify active targets, isolated a selector mismatch, and patched the service manifest."*
- **Result & Guardrail:** Conclude with metrics and preventive automation: *"Traffic recovered immediately; I added automated pre-merge manifest validation in GitHub Actions to prevent untracked label drift."*

4. The 60-Second Command Toolkit

```
# 1. Timeline of recent cluster failures
kubectl get events -A --sort-by=.metadata.creationTimestamp | tail -n 20

# 2. Crash forensics: previous container stderr
kubectl logs <pod-name> --previous --tail=100

# 3. Node allocatable capacity vs current usage
kubectl describe node <node-name> | grep -A 8 "Allocatable"

# 4. Ephemeral storage unlinked open file handles
lsof +L1
```