SRE Kubernetes Scenario-Based Questions and Answers

1. 500 Errors from Kubernetes-hosted API

- - Step 1: Identify affected pods via logs:

kubectl logs -l app=my-api --tail=100

- - Step 2: Check pod health and events:

kubectl describe pod <pod-name>

- - Step 3: Inspect service and ingress:

kubectl get svc,ingress -n <namespace>

- - Step 4: Correlate with Prometheus/Grafana for spikes.
- - Remediation: Fix code/config bug, rollout patch:

kubectl rollout restart deployment my-api

2. Job Failed Silently

- - Add observability:
- - Use Kubernetes CronJob with .spec.successfulJobsHistoryLimit
- - Add alerting via Prometheus + Alertmanager:
- - alert: JobFailed

expr: kube_job_status_failed > 0

for: 5m

labels:

severity: critical

- - Log to centralized system (e.g., Loki/ELK).

3. OOMKilled Pods

- - Check OOM events:

kubectl get events --field-selector reason=OOMKilling

- - Right-size limits using metrics from kubectl top pod or Prometheus:

resources:

requests:

memory: "256Mi"

limits:

memory: "512Mi"

- - Use VPA (Vertical Pod Autoscaler) and monitor with kube-state-metrics.

4. Safe Deployment Strategies

- - Implement:
- - RollingUpdate with pause:

strategy:

type: RollingUpdate

rollingUpdate:

maxUnavailable: 1

maxSurge: 1

- - Canary using Flagger or Argo Rollouts
- - Manual promotion pipeline in CI/CD (e.g., GitHub Actions, Azure DevOps).

5. Increased Latency Between Microservices

- - Tools:
- - Istio/Linkerd metrics
- - Prometheus + Grafana dashboard
- - kubectl exec with curl and time for latency
- - Check:
- - Service discovery
- - DNS latency
- - Network policies
- - Resource pressure on nodes

6. Multi-Region DR for Stateful Workload

- - Steps:

- - Use etcd backups or storage replication (e.g., Portworx, Velero)
- - Deploy to second region with same manifests
- - Test failover with simulated outages
- - Test DR:

kubectl cordon node && kubectl drain node --ignore-daemonsets

7. Node Flapping Ready/NotReady

- - Investigate:

kubectl describe node <node-name>

journalctl -u kubelet

- - Common causes:
- - Resource exhaustion
- - Kubelet crash
- - CNI issues
- - Prevent:
- - Use node auto-replacement with autoscaler
- - Monitor node health

8. Custom Resource Reconcile Failure

- - Check:

kubectl get crd

kubectl describe <custom-resource>

kubectl logs -l name=<operator-name>

- - Fix:
- - Validate CRD versions
- - Restart operator or reconcile manually

9. Pod Evictions in Node Pool

- - Check:

kubectl get events | grep Evicted

- - Investigate node pressure (kubectl describe node)
- - Causes:
- - Disk pressure

- - Memory shortage
- - Prevent:
- - Taints and tolerations
- - Resource requests/limits
- - Monitor with kubelet exporter

10. DNS Failures (CoreDNS)

- - Debug:

kubectl logs -n kube-system -l k8s-app=kube-dns

kubectl exec -it <pod> -- nslookup <svc-name>

- - Tune:
- - Increase CoreDNS replicas
- - Cache TTL
- - Monitor:

Use CoreDNS metrics plugin + Prometheus

11. Prometheus Scrape Intermittent

- - Check targets:

http://<prometheus>:9090/targets

- - HA setup:

Use Thanos or Cortex

- - Fix:

Adjust scrape_interval, service monitor timeout

12. Prometheus vs App Metrics Discrepancy

- - Validate app endpoint manually:

curl http://<pod-ip>:<port>/metrics

- - Check exporter logic or label mismatch
- - Prometheus relabel config validation

13. Pod in ImagePullBackOff

- - Advanced checks:

kubectl describe pod <pod>

docker pull <image> on node (if accessible)

- - Check:
- - Image secret
- - Network/DNS
- - Tag existence

14. Audit Kubernetes for SRE Compliance

- - Focus Areas:
- - RBAC policies
- - Resource limits
- - Network Policies
- - Pod disruption budgets
- - Logging and monitoring presence
- - CI/CD pipelines

15. Scaling But Dropped Traffic

- - Check:
- - HPA status
- - Service endpoints: kubectl get ep
- - Load balancer capacity
- - Fix:
- - Pre-warming
- - Use Istio gateway load testing

16. CPU Throttling Alerts

- - Check metrics:

container_cpu_cfs_throttled_seconds_total

- - Fix:
- - Increase CPU limits
- - Use guaranteed QoS class (request = limit)

17. Network Policy Blocked Services

- - Detect:
- - Use Calico or Cilium observability
- - kubectl get networkpolicy
- - Prevent:
- - Alert on blocked connections
- - Canary policy testing

18. Ingress Route DNS Issues

- - Debug:

kubectl describe ingress

nslookup <domain>

dig +trace <domain>

- - Check:
- - DNS propagation
- - TLS certs

19. PVC Stuck in Pending

- - Check:

kubectl describe pvc <name>

kubectl get storageclass

- - Fix:
- - Ensure provisioner is installed
- - Permissions correct

20. High API Server Latency

- - Check metrics:

apiserver_request_duration_seconds

- - Scale:
- - Add control plane nodes (in self-managed)
- - Tune etcd and kube-apiserver flags
- - Offload controller load