Rakshita Belwal

Kubernetes Scenario Based Interview Questions with Expected Answers

Scenario 1: A Pod Keeps Crashing After Deployment

Question:
You've deployed a Pod, but it keeps crashing. How would you troubleshoot the issue?

Expected Answer:

- Check Pod status: `kubectl get pods`
- Inspect logs: `kubectl logs <pod_name>`
- Describe the Pod for events/errors: `kubectl describe pod <pod_name>`
- Execute into the Pod to debug: `kubectl exec -it <pod_name> -- /bin/sh`
- Validate environment variables and configurations

Scenario 2: High Traffic Causes Pod Failures

Question:
Your Kubernetes service is experiencing high traffic, and some requests are failing. How do you handle this?

Expected Answer:

- Scale the deployment: `kubectl scale deployment <name> --replicas=5`

- Use HPA (Horizontal Pod Autoscaler): `kubectl autoscale deployment <name> --min=2 --max=10 --cpu-percent=80`
- Ensure sufficient cluster resources using `kubectl top nodes`
- Load balance traffic with an Ingress controller or Service

Scenario 3: A Service is Not Reachable

Question:

A deployed service is not accessible from outside the cluster. What could be wrong?

Expected Answer:

- Verify if the service type is ClusterIP, NodePort, or LoadBalancer
- Check the exposed ports using `kubectl get svc <service_name>`
- Ensure the service selector matches the pod labels

- Use `kubectl port-forward` for debugging

Scenario 4: Persistent Volume Not Mounting Properly

Question:

You have configured a Persistent Volume (PV) and Persistent Volume Claim (PVC), but the Pod is unable to mount it. What do you do?

Expected Answer:

- Check PVC status: `kubectl get pvc`
- Verify if PV is correctly bound to PVC
- Ensure the **storage class is supported** by the cluster
- Describe the Pod and check for mount-related errors

Scenario 5: A Node Becomes Unavailable

Question:

One of the Kubernetes worker nodes suddenly becomes unavailable. What actions should you take?

Expected Answer:

- Check node status: `kubectl get nodes`
- Inspect node conditions: `kubectl describe node <node_name>`
- Investigate kubelet logs on the node
- Drain and remove the node if necessary: `kubectl drain <node_name> --ignore-daemonsets --delete-local-data`
- Ensure Pods are scheduled on healthy nodes

Scenario 6: Blue-Green Deployment in Kubernetes

Question:

How would you implement a Blue-Green Deployment in Kubernetes?

Expected Answer:

- Deploy a **new version (Green) alongside the old one (Blue)**
- Use **Service and Labels** to switch traffic
- Test the Green version, then update the service selector to point to it
- Roll back if needed using previous deployment history

Rakshita Belwal

Scenario 7: Securing Kubernetes Secrets

Question:

How can you securely manage sensitive credentials (API keys, passwords) in Kubernetes?

Expected Answer:

- Use Kubernetes **Secrets** instead of ConfigMaps
- Store secrets in a secret management tool (e.g., HashiCorp Vault, AWS Secrets Manager)
- Mount secrets as environment variables or files
- Restrict access to Secrets using RBAC

Scenario 8: CI/CD Pipeline Deployment with Kubernetes

Question:

How would you automate deployments in Kubernetes using CI/CD?

Expected Answer:

- Use a **CI/CD tool** like Jenkins, GitHub Actions, or ArgoCD
- Build and push Docker images on code changes
- Apply manifests via `kubectl apply -f` or Helm
- Implement **Rolling Updates** for zero downtime

Scenario 9: A Rolling Update Fails Midway

Question:

You applied a Rolling Update, but some Pods failed, causing an inconsistent state. What's your approach to fixing this?

Expected Answer:

- Check rollout status: `kubectl rollout status deployment <deployment_name>`
- View previous revisions: `kubectl rollout history deployment <deployment_name>`
- Roll back if needed: `kubectl rollout undo deployment <deployment_name>`
- Debug failing Pods: `kubectl describe pod <pod_name>`

Scenario 10: A Pod is Stuck in 'Pending' State

Question:

A Pod remains in the "Pending" state for too long. What do you investigate?

Expected Answer:

- Check `kubectl describe pod <pod_name>` for scheduling issues

- Verify node resources: `kubectl get nodes --show-labels`
- Inspect Events: `kubectl get events --sort-by=.metadata.creationTimestamp`
- If it's a volume issue, check `kubectl get pvc`

Scenario 11: Kubernetes Service is Not Routing Traffic Correctly

Question:

You deployed a service, but it’s not distributing traffic to the Pods. What could be the issue?

Expected Answer:

- Check service selectors and Pod labels match
- Ensure Pods are in "Running" state and have valid IPs
- Validate service endpoints: `kubectl get endpoints <service_name>`
- If using Ingress, check rules with `kubectl describe ingress`

Scenario 12: How Would You Handle a Multi-Cluster Deployment?

Question:

Your company needs to deploy Kubernetes workloads across multiple clusters. How would you approach it?

Expected Answer:

- Use **KubeFed (Kubernetes Federation)** for cross-cluster management
- Leverage tools like **ArgoCD** or **FluxCD** for GitOps-based multi-cluster deployments
- Use **Service Mesh (Istio/Linkerd)** to manage inter-cluster communication

---

Scenario 13: A Pod is Restarting Frequently

Question:

A Pod is constantly restarting. How do you troubleshoot?

Expected Answer:

- Check logs: `kubectl logs <pod_name> --previous`
- Describe the Pod: `kubectl describe pod <pod_name>`
- Investigate Liveness/Readiness probes in deployment YAML
- Verify resource limits (`kubectl get nodes` to check resource pressure)

Scenario 14: DNS Resolution is Not Working in Kubernetes

Question:

Your application Pods can't resolve DNS names of services inside the cluster. What do you do?

Expected Answer:

- Check CoreDNS status: `kubectl get pods -n kube-system | grep coredns`
- Test DNS resolution: `kubectl exec -it <pod_name> -- nslookup <service_name>`
- Verify service discovery settings in `/etc/resolv.conf` inside the Pod
- Restart CoreDNS if necessary

Scenario 15: High Latency in Kubernetes Applications

Question:

Users report that your Kubernetes-based application is experiencing high latency. What's your debugging approach?

Expected Answer:

- **Check resource usage** with `kubectl top nodes` and `kubectl top pods`

- **Analyze network policies** (`kubectl get networkpolicy`)
- **Enable distributed tracing** with Jaeger/OpenTelemetry
- **Check Ingress controller logs** for bottlenecks

Rakshita Belwal