Kubernetes Interview Questions & Troubleshooting Answers

A Practical Handbook for DevOps Job Seekers

1. What happens when a pod is stuck in ContainerCreating?

Purpose: To evaluate your knowledge of the pod lifecycle and your ability to investigate container startup problems.

Context:

This usually occurs when the scheduler has assigned a node, but the container runtime cannot finish setting up the container. Common causes include image pull issues, missing secrets/configs, or volume mount delays.

Troubleshooting Answer:

1. **kubectl describe pod <pod-name> # Check events**
2. **kubectl get pv,pvc   # Check volume bindings**
3. **kubectl logs <pod-name>   # See if image pull is the issue**

Look for: ImagePullBackOff, volume mount issues, network errors.

2. How do you handle a CrashLoopBackOff error?

Purpose: To assess your troubleshooting ability for runtime and application configuration failures.

Context:

Indicates the pod started, crashed due to an application issue, and is now in a restart loop. Often seen with misconfigured environment variables or bad startup commands.

Troubleshooting Answer:

1. **kubectl logs <pod-name> --previous**
2. **kubectl describe pod <pod-name>**
3. **kubectl get configmap,secret**

Look for: Misconfigured probes, environment issues, missing dependencies.

3. A node is in NotReady state. What steps do you take?

Purpose: To confirm your understanding of cluster health and how to diagnose node-level issues.

Context:

Nodes go NotReady when the kubelet is not reporting healthy status to the API server, often due to resource pressure or network failures.

Troubleshooting Answer:

1. **kubectl get nodes**
2. **ssh <node> "sudo systemctl status kubelet"**
3. **df -h # Check disk space**

Look for: Kubelet failures, disk/memory saturation, network unreachable.

4. DNS resolution is failing inside a pod. How will you fix it?

Purpose: To verify your knowledge of internal DNS resolution and network policy behavior within the cluster.

Context:

Internal or external names can’t be resolved inside pods. This usually relates to CoreDNS failure, a broken **resolv.conf**, or network policy issues.

Troubleshooting Answer:

1. **kubectl exec -it <pod> -- nslookup google.com**

2. **kubectl get pods -n kube-system | grep coredns**

3. **kubectl logs <coredns-pod> -n kube-system**

Look for: CoreDNS pod failures, incorrect network policies.

5. Your service is not reachable via ClusterIP. Why?

Purpose: To check your understanding of Kubernetes service discovery and endpoint bindings.

Context:

A service created with ClusterIP is not routing traffic to backend pods. Often due to incorrect selectors or port mismatches between service and pod.

Troubleshooting Answer:

1. **kubectl describe svc <svc-name>**

2. **kubectl get endpoints <svc-name>**

3. **kubectl get pods -l app=<label>**

Look for: Label mismatches, port mismatches, missing endpoints.

6. Why would a PVC be stuck in Pending?

Purpose: To evaluate your understanding of persistent storage provisioning and storage class configuration.

Context:

A PersistentVolumeClaim can’t bind to a suitable PersistentVolume due to storage class mismatches or no available capacity.

Troubleshooting Answer:

```
1. kubectl get pvc

2. kubectl describe pvc <pvc-name>

3. kubectl get sc  # Check storage class
```

Look for: No matching PV, misconfigured storage class.

7. kubectl exec and logs are hanging. How do you troubleshoot?

Purpose: To confirm your ability to diagnose communication problems between control plane and nodes.

Context:

This typically indicates networking issues between the control plane and the kubelet, or the container runtime is unresponsive.

Troubleshooting Answer:

```
1. kubectl describe pod <pod-name>

2. kubectl get pod -o wide  # Get IP

3. kubectl top node         # Check resource pressure
```

Look for: Network issues, kubelet health, resource bottlenecks.

8. Ingress is not routing traffic. What do you do?

Purpose: To assess your understanding of traffic routing, Ingress configuration, and backend service connectivity.

Context:

Ingress is configured, but requests are not reaching the backend. May be caused by incorrect host/path settings or broken Ingress controller setup.

Troubleshooting Answer:

1. **kubectl describe ingress <name>**

2. **kubectl logs <ingress-controller-pod>**

3. **kubectl get svc -n ingress-nginx**

Look for: DNS mismatch, backend unreachable, wrong path/host rules.

9. Why is HPA not scaling your pods?

Purpose: To determine your awareness of autoscaling requirements and monitoring setup in Kubernetes.

Context:

HPA relies on metrics-server to retrieve CPU/memory usage. If metrics are missing or below thresholds, no scaling occurs.

Troubleshooting Answer:

1. **kubectl get hpa**
2. **kubectl top pods**
3. **kubectl get pods -n kube-system | grep metrics-server**

Look for: No metrics-server, thresholds not met, misconfigured CPU/memory targets.

10. You updated a Deployment but pods didn’t change. Why?

Purpose: To test your ability to manage deployment strategies and version control for applications.

Context:

Often happens when the manifest has no actual spec change or uses an unchanged image tag. Kubernetes sees no diff and doesn’t roll out.

Troubleshooting Answer:

1. **kubectl rollout status deployment <name>**
2. **kubectl rollout history deployment <name>**
3. **kubectl describe deployment <name>**

Look for: No actual spec change, same image tag used, caching.

Found this useful?

Follow

Let's connect

💬 Found it useful? Drop your thoughts below and share it with your fellow DevOps engineers!

Follow me for more DevOps, Kubernetes, and cloud-native insights.

🔁 **Repost**