Kubernetes interview question and answers

1. What is Kubernetes and how does it differ from Docker?

Answer:
Kubernetes is an open-source container orchestration system for automating deployment, scaling, and management of containerized applications. It manages clusters of machines and schedules containers onto them, handling load balancing, scaling, and self-healing.

Docker is a platform and runtime for creating and running containers. It focuses on packaging applications into containers but does not provide orchestration.

Kubernetes can use Docker as a container runtime but adds orchestration, multi-node cluster management, and declarative management capabilities. Docker solves containerization, Kubernetes solves orchestration.

---

2. Explain Kubernetes Pods and why they are the smallest deployable units.

Answer:
A Pod is the smallest deployable unit in Kubernetes, representing one or more containers sharing the same network namespace, storage volumes, and lifecycle. Containers in a Pod are tightly coupled and run on the same node.

Pods are ephemeral and managed by higher-level controllers like ReplicaSets or Deployments to ensure availability and scaling. Pods abstract the container runtime and provide a manageable unit for Kubernetes to schedule and manage.

---

3. What is a Deployment and how does it help manage application updates?

Answer:
A Deployment manages ReplicaSets and Pods declaratively. It defines the desired state for application instances, including the number of replicas and container image.

Deployments enable rolling updates by incrementally replacing Pods with new versions. It monitors health and readiness, ensuring minimal downtime during updates. If failures occur, Deployments allow pausing or rolling back to previous stable versions.

---

4. How does Kubernetes handle rolling updates to avoid downtime?

Answer:
Kubernetes updates Pods incrementally by creating new Pods with the updated specification while gradually terminating old Pods. Parameters like maxSurge and maxUnavailable control how many Pods can be created or removed simultaneously.

It waits for readiness probes to pass on new Pods before removing old ones, ensuring continuous availability. If readiness fails, the rollout pauses for intervention or rollback.

---

5. Scenario: A Pod remains in Pending status indefinitely. What could cause this and how do you fix it?

Answer:
Pending means the Pod is accepted but not scheduled. Causes:

- Insufficient node resources (CPU, memory) to fulfill Pod requests.
- Nodes marked Unschedulable or NotReady.
- Node taints without corresponding Pod tolerations.
- Scheduling constraints like affinity/anti-affinity rules not met.

Troubleshooting:

- kubectl describe pod <pod> shows events and failure reasons.
- Check node resources: kubectl describe nodes or cluster metrics.
- Check node taints: kubectl describe nodes | grep Taints.
- Adjust resource requests, add tolerations, or scale cluster nodes.

---

6. What is the difference between a ReplicaSet and a ReplicationController?

Answer:
ReplicationController ensures a specified number of Pods are running using equality-based selectors. ReplicaSet is the newer version with support for set-based label selectors (in, notin), offering more flexibility.

ReplicaSet is the recommended controller now, and Deployments use ReplicaSets internally. ReplicationControllers are deprecated and mainly for legacy support.

---

7. How do Services provide stable networking for Pods?

Answer:
Pods have dynamic IPs and can be ephemeral. A Service creates a stable virtual IP and DNS name that load balances traffic to matching Pods using label selectors.

Kube-proxy configures routing rules to forward traffic to healthy Pod endpoints, abstracting Pod lifecycle changes from clients.

---

8. Scenario: Your Service's Endpoints list is empty. What might be wrong?

Answer:
Empty endpoints mean no Pods match the Service selector or Pods are not ready.

Check:

- Pod labels and Service selector match: kubectl get pods --show-labels and Service YAML.
- Pod readiness probes: Pods failing readiness are excluded.
- Network policies or firewall blocking traffic.
- Pod status and logs for errors preventing readiness.

---

9. What are Namespaces and why are they used?

Answer:
Namespaces provide logical separation within a cluster to isolate resources, users, and policies. Useful for multi-tenancy, separating dev/test/prod environments, or multiple teams sharing a cluster.

Namespaces enable resource quotas, scoped RBAC, and prevent name collisions.

---

10. Explain what an Ingress resource is and how it differs from a LoadBalancer service.

Answer:
Ingress manages external HTTP/S access to multiple services via a single IP and DNS. It provides path-based or host-based routing and TLS termination.

LoadBalancer exposes a single service externally with its own IP, typically via cloud provider integration.

Ingress requires an Ingress Controller to implement routing logic and is more cost-effective for multi-service routing.

---

11. How does Kubernetes handle self-healing of Pods?

Answer:
Kubernetes monitors Pod health via liveness and readiness probes. If a Pod becomes unresponsive or fails health checks, the kubelet restarts the container or the controller replaces the Pod.

Controllers ensure desired replica counts by creating new Pods when failures occur, maintaining availability.

---

12. What are readiness probes and how do they affect Service traffic routing?

Answer:
Readiness probes check if a container is ready to serve traffic. Pods failing readiness probes are excluded from Service endpoints, preventing routing to unhealthy Pods.

This prevents users from hitting Pods that are starting up or in error states, improving availability.

---

13. Scenario: Your Deployment rollout is stuck with new Pods failing readiness probes. How do you troubleshoot?

Answer:
Steps:

- Check Pod logs for errors.
- Verify readiness probe configuration (HTTP path, port).
- Confirm application inside Pod starts correctly.
- Check resource limits causing startup delays.
- Use kubectl describe pod for events.
- Roll back Deployment if needed to restore service.

---

14. What is a ReplicaSet's role in scaling applications?

Answer:
ReplicaSets maintain a stable set of replicas. To scale, you increase or decrease the replica count in the ReplicaSet or Deployment, prompting Kubernetes to add or remove Pods accordingly.

Autoscalers can automate this based on CPU/memory or custom metrics.

---

15. What is the difference between a ClusterIP, NodePort, and LoadBalancer Service?

Answer:

- **ClusterIP:** Default; exposes service internally inside the cluster.
- **NodePort:** Exposes service on a static port on each node's IP for external access.
- **LoadBalancer:** Provisions a cloud provider load balancer with external IP.

Use depends on exposure needs.

---

16. Explain how network policies control traffic in Kubernetes.

Answer:
NetworkPolicies define rules to allow or deny traffic between Pods or from external sources, controlling ingress and egress at the IP and port level.

They enhance security by isolating workloads and enforcing least privilege.

---

17. Scenario: Pods in your namespace cannot communicate with each other despite being on the same network. What could be wrong?

Answer:
Likely causes:

- Network policies blocking traffic.
- CNI plugin misconfiguration or failure.
- Firewall rules or cloud security groups restricting pod network.
- Pod misconfiguration (wrong ports, protocols).

Check applied NetworkPolicies, CNI logs, and Pod configurations.

---

18. What are labels and selectors, and how do they work together?

Answer:
Labels are key-value pairs attached to Kubernetes objects for identification. Selectors query these labels to group and select objects.

Selectors power Services, ReplicaSets, and other controllers to manage sets of Pods.

---

19. How does kube-proxy facilitate Service traffic routing?

Answer:
Kube-proxy runs on each node and manages virtual IPs and iptables/ipvs rules to forward incoming Service traffic to backend Pods.

It abstracts routing complexity and balances load across Pods.

---

20. What is a Deployment rollback and how is it performed?

Answer:
A rollback restores a Deployment to a previous stable revision if an update causes failures.

Use kubectl rollout undo deployment/<name> to revert. Kubernetes switches back to the prior ReplicaSet, replacing failing Pods with the stable ones.

---

21. Scenario: Your Pods crash-loop after a Deployment update. What debugging steps do you follow?

Answer:

- Check Pod logs with kubectl logs.
- Inspect container command, arguments, and environment variables.
- Validate configMaps and secrets mounted correctly.
- Review resource requests/limits.
- Examine Deployment spec for errors.
- Rollback if needed.

22. What is a ReplicationController and why is it considered legacy?

Answer:
ReplicationController is the original controller to maintain a specified number of Pod replicas but supports only equality-based selectors.

It’s replaced by ReplicaSet, which supports richer selectors and is managed by Deployments.

---

23. What is the role of the kube-scheduler?

Answer:
The scheduler assigns newly created Pods to nodes based on resource availability, taints/tolerations, affinity rules, and other constraints.

It ensures Pods are placed optimally within the cluster.

---

24. How do you manage persistent storage in Kubernetes?

Answer:
Use PersistentVolumes (PV) and PersistentVolumeClaims (PVC) to abstract storage provisioning and consumption.

Storage classes allow dynamic provisioning. Volumes attach to Pods, surviving Pod restarts and scaling.

---

25. Scenario: PVC is stuck in Pending. What does it mean and how to fix it?

Answer:
Pending PVC means no suitable PersistentVolume is available or provisioning failed.

Check:

- StorageClass correctness.

- Available PV capacity and access modes.
- Cloud provider quotas.
- Provisioner logs.

Correct errors or create matching PV.

---

26. Explain how Ingress controllers work.

Answer:
Ingress controllers watch Ingress resources and configure load balancers or proxies (like NGINX) to implement routing rules.

They translate Ingress specs into network configuration, enabling centralized HTTP/S access management.

---

27. What is the difference between liveness and readiness probes?

Answer:

- **Liveness probes** detect if a container is alive; failing liveness triggers a restart.
- **Readiness probes** detect if a container is ready to serve traffic; failing readiness excludes it from Services.

Both improve app resilience and availability.

---

28. How would you secure Kubernetes clusters?

Answer:
Best practices:

- Enable RBAC with least privilege.
- Use NetworkPolicies.
- Secure etcd and API server access.
- Use Pod Security Policies or OPA Gatekeeper.
- Enable TLS for all components.
- Scan images for vulnerabilities.

29. Scenario: Users report intermittent failures accessing your app. How do you diagnose?

Answer:
Steps:

- Check Pod logs and events for errors.
- Inspect Service and Ingress logs.
- Verify resource utilization for throttling.
- Confirm network policies and firewall rules.
- Use monitoring and tracing tools.

---

30. What is a StatefulSet and how does it differ from a Deployment?

Answer:
StatefulSet manages Pods with stable network IDs and persistent storage, ordered deployment and scaling.

Used for stateful applications like databases.

Deployments manage stateless apps without stable identity or persistent volume guarantees.

---

31. Explain Horizontal Pod Autoscaler (HPA).

Answer:
HPA scales Pods in/out based on observed metrics (CPU, memory, custom).

It queries metrics API and adjusts Deployment or ReplicaSet replicas to meet demand.

---

32. Scenario: Your HPA is not scaling Pods despite CPU usage rising. How to troubleshoot?

Answer:
Check:

- Metrics-server installed and working.
- Correct resource requests set on Pods.
- HPA configured with correct metrics and thresholds.
- kubectl describe hpa for events.
- Metrics API accessibility.

---

33. How do you upgrade a Kubernetes cluster safely?

Answer:
Steps:

- Backup cluster state.
- Upgrade master/control plane nodes first.
- Then upgrade worker nodes, draining nodes before upgrade.
- Test applications after upgrade.
- Rollback if needed.

Follow cloud provider or distro-specific guides.

---

34. What is Kops and when would you use it?

Answer:
Kops is a tool for provisioning and managing Kubernetes clusters on AWS and similar clouds.

It automates cluster creation, upgrades, and management, ideal for production clusters on cloud infrastructure.

---

35. Scenario: After upgrading your cluster with Kops, some nodes fail to join. What do you do?

Answer:
Check:

- Node logs for kubelet errors.
- Network configurations and security groups.
- IAM permissions.
- Kops cluster state consistency.
- Rolling upgrade progress.

---

36. How do affinity and anti-affinity rules affect Pod scheduling?

Answer:
Affinity rules prefer or require Pods to be co-located or separated on nodes or zones for performance, fault tolerance, or compliance.

Anti-affinity ensures Pods do not schedule together, increasing availability.

---

37. Explain the role of etcd in Kubernetes.

Answer:
etcd is a distributed key-value store holding Kubernetes cluster state.

It stores configurations, resource metadata, and is critical for cluster consistency and operation.

---

38. Scenario: Your API server is unreachable. What could be wrong?

Answer:
Possible causes:

- Network outage.
- API server crash or pod failure.
- Authentication/authorization issues.
- etcd failure.

- Certificate expiration.

Check logs, endpoints, network, and cluster components.

---

39. What is a DaemonSet and where would you use it?

Answer:
DaemonSet runs a copy of a Pod on each node or subset of nodes.

Used for logging, monitoring agents, or node-level utilities requiring presence on every node.

---

40. How does Kubernetes implement service discovery?

Answer:
Through Services exposing a stable DNS name and ClusterIP.

CoreDNS resolves Service names to IPs, enabling Pods to communicate using service names rather than IPs.

---

41. Scenario: You want to restrict which Pods can run on a node. How?

Answer:
Use node taints to repel Pods and tolerations on Pods to allow scheduling.

Use node selectors and affinity rules to constrain Pods to specific nodes.

---

42. Explain the difference between ephemeral and persistent storage in Kubernetes.

Answer:
Ephemeral storage is tied to Pod lifecycle and lost when Pod dies.

Persistent storage survives Pod restarts and can be shared or retained, via PVs and PVCs.

43. How do you debug network connectivity issues between Pods?

Answer:
Use:

- kubectl exec to run network tools inside Pods.
- Check NetworkPolicy rules.
- Verify CNI plugin health.
- Review firewall or security groups.
- Use tcpdump or tracing tools.

---

44. What is the role of kubelet?

Answer:
Kubelet runs on each node, managing Pods lifecycle, reporting status to the API server, and ensuring containers run as expected.

---

45. Scenario: Your Pods are restarting frequently. How do you investigate?

Answer:
Check:

- Pod logs for errors.
- Resource limits causing OOM kills.
- CrashLoopBackOff reasons with kubectl describe pod.
- Readiness and liveness probe misconfigurations.
