**Kubernetes Interview Questions - Study Notes (Day 36)**

**Course: Complete DevOps Course**  
**Instructor: Abhishek**

**1. Top 10 Kubernetes Interview Questions**

**Q1: Difference Between Docker and Kubernetes?**

✅ **Docker:**

* Runs single containers.
* Single-host architecture.
* No built-in auto-healing or scaling.

✅ **Kubernetes:**

* Manages container clusters.
* Multi-host, HA architecture.
* Auto-healing, scaling, and advanced features.

📌 **Key Point:** Kubernetes abstracts and orchestrates Docker containers.

**Q2: Main Components of Kubernetes Architecture?**

**Control Plane (Master Node):**

* **API Server:** Handles all requests.
* **Scheduler:** Assigns Pods to nodes.
* **etcd:** Stores cluster state.
* **Controller Manager:** Manages controllers.
* **Cloud Controller Manager:** Integrates with cloud providers.

**Data Plane (Worker Nodes):**

* **Kubelet:** Manages Pod lifecycle.
* **Kube-Proxy:** Handles networking.
* **Container Runtime:** Runs containers (Docker, containerd).

**Q3: Docker Swarm vs. Kubernetes?**

| **Feature** | **Docker Swarm** | **Kubernetes** |
| --- | --- | --- |
| **Complexity** | Simple setup | Enterprise-grade |
| **Scaling** | Limited | Advanced (HPA, Autoscaler) |
| **Networking** | Basic | Advanced (CNI plugins) |
| **Ecosystem** | Limited tools | Rich ecosystem |

📌 **Use Case:** Swarm for small apps; Kubernetes for large-scale production.

**Q4: Docker Container vs. Kubernetes Pod?**

✅ **Docker Container:** Runs via docker run (ephemeral).  
✅ **Kubernetes Pod:** Defined in YAML, supports multiple containers.

apiVersion: v1

kind: Pod

metadata:

name: nginx

spec:

containers:

- name: nginx

image: nginx:1.14.2

**Q5: What is a Namespace?**

* **Logical isolation** of resources.
* Used for multi-team environments, RBAC, and network policies.

kubectl create namespace dev

kubectl get pods -n dev

**Q6: Role of Kube-Proxy?**

* Manages networking rules on nodes.
* Uses **IP tables/IPVS** to route traffic:
  + Service → Pod (load balancing).
  + External → NodePort/LoadBalancer.

**Q7: Types of Kubernetes Services?**

| **Type** | **Accessibility** | **Use Case** |
| --- | --- | --- |
| **ClusterIP** | Internal cluster | Microservices |
| **NodePort** | Worker node IPs | Internal access |
| **LoadBalancer** | Public (cloud) | External access |

apiVersion: v1

kind: Service

metadata:

name: nginx-service

spec:

type: LoadBalancer

ports:

- port: 80

selector:

app: nginx

**Q8: NodePort vs. LoadBalancer?**

* **NodePort:** Exposes on worker node IPs (<NodeIP>:30007).
* **LoadBalancer:** Creates cloud LB (e.g., AWS ELB).

**Q9: Role of Kubelet?**

* Runs on **worker nodes**.
* Ensures Pods are healthy (auto-restarts if needed).
* Reports Pod status to API Server.

📌 **Key Task:** Monitors Pods and triggers auto-healing.

**Q10: Day-to-Day Kubernetes Tasks?**

1. **Cluster Maintenance:** Upgrades, security patches.
2. **Deployments:** Rolling updates, rollbacks.
3. **Troubleshooting:** Debugging Pods (kubectl describe/logs).
4. **Monitoring:** Setting up Prometheus/Grafana.
5. **RBAC/Networking:** Managing access, network policies.

**2. Pro Tips for Interviews**

* **Practice Commands:**
* kubectl get pods -o wide
* kubectl describe svc <name>
* kubectl logs <pod-name>
* **Understand YAML:** Know deployment.yaml, service.yaml.
* **Real-World Examples:** Relate answers to past projects.

**3. Summary**

✅ Mastered **10 key Kubernetes interview questions**.  
✅ Learned architecture, services, deployments, and real-world tasks.  
✅ Ready for troubleshooting, scaling, and production environments.

🚀 **Next:** Kubernetes Ingress & Helm!

📢 **Feedback?** Comment below! 👍 **Like & Share** if this helped!

**End of Notes** 🎉

# \*\*Kubernetes Interview Questions - Study Notes (Day 36)\*\*

\*\*Course: Complete DevOps Course\*\*

\*\*Instructor: Abhishek\*\*

---

## \*\*1. Top 10 Kubernetes Interview Questions\*\*

### \*\*Q1: Difference Between Docker and Kubernetes?\*\*

✅ \*\*Docker:\*\*

- Container platform (runs single containers).

- Single-host architecture.

- No built-in auto-healing/scaling.

✅ \*\*Kubernetes:\*\*

- Container \*\*orchestration\*\* platform (manages clusters).

- Multi-host, HA architecture.

- Auto-healing, scaling, enterprise features (load balancing, service discovery).

📌 \*\*Key Point:\*\* Kubernetes adds \*\*abstraction layers\*\* (Pods, Deployments) over Docker containers.

---

### \*\*Q2: Main Components of Kubernetes Architecture?\*\*

#### \*\*Control Plane (Master Node):\*\*

- \*\*API Server:\*\* Entry point for all requests.

- \*\*Scheduler:\*\* Assigns Pods to nodes.

- \*\*etcd:\*\* Key-value store (cluster state).

- \*\*Controller Manager:\*\* Manages controllers (e.g., ReplicaSet).

- \*\*Cloud Controller Manager:\*\* Integrates with cloud providers (e.g., AWS ELB).

#### \*\*Data Plane (Worker Nodes):\*\*

- \*\*Kubelet:\*\* Manages Pod lifecycle.

- \*\*Kube-Proxy:\*\* Handles networking (IP tables/IPVS).

- \*\*Container Runtime:\*\* Runs containers (Docker, containerd).

![K8s Architecture](https://d33wubrfki0l68.cloudfront.net/2475489eaf20163ec0f54ddc1d92aa8d4c87c96b/e7c81/images/docs/components-of-kubernetes.svg)

---

### \*\*Q3: Docker Swarm vs. Kubernetes?\*\*

| \*\*Feature\*\* | \*\*Docker Swarm\*\* | \*\*Kubernetes\*\* |

|-------------------|--------------------------------|---------------------------------|

| \*\*Complexity\*\* | Simple setup | Complex (enterprise-grade) |

| \*\*Scaling\*\* | Limited | Advanced (HPA, Cluster Autoscaler)|

| \*\*Networking\*\* | Basic (overlay networks) | Advanced (CNI plugins like Calico)|

| \*\*Ecosystem\*\* | Limited third-party tools | Rich (CNCF projects, Helm, etc.)|

📌 \*\*Use Case:\*\* Swarm for small apps; Kubernetes for production-scale.

---

### \*\*Q4: Docker Container vs. Kubernetes Pod?\*\*

✅ \*\*Docker Container:\*\*

- Runs via `docker run -d -p 80:80 nginx`.

- Ephemeral (no auto-healing).

✅ \*\*Kubernetes Pod:\*\*

- Defined in `pod.yaml` (declarative).

- Can run \*\*1+ containers\*\* (shared networking/storage).

- Managed by Deployments (auto-healing).

```yaml

# Example Pod

apiVersion: v1

kind: Pod

metadata:

name: nginx

spec:

containers:

- name: nginx

image: nginx:1.14.2

```

---

### \*\*Q5: What is a Namespace?\*\*

- \*\*Logical isolation\*\* of resources in a cluster.

- Used for:

- Multi-team environments (e.g., `dev`, `prod`).

- RBAC (Role-Based Access Control).

- Network policies.

```sh

kubectl create namespace dev

kubectl get pods -n dev # List Pods in 'dev' namespace

```

---

### \*\*Q6: Role of Kube-Proxy?\*\*

- Manages \*\*networking rules\*\* on worker nodes.

- Uses \*\*IP tables/IPVS\*\* to route traffic:

- Service → Pod (load balancing).

- External → NodePort/LoadBalancer.

📌 \*\*Example:\*\* Routes traffic from `NodeIP:30007` to a Pod.

---

### \*\*Q7: Types of Kubernetes Services?\*\*

| \*\*Type\*\* | \*\*Accessibility\*\* | \*\*Use Case\*\* |

|----------------|---------------------------------|----------------------------------|

| \*\*ClusterIP\*\* | Internal (within cluster) | Inter-microservice communication |

| \*\*NodePort\*\* | Via worker node IPs (VPC) | Internal testing |

| \*\*LoadBalancer\*\*| Public (cloud providers) | Internet-facing apps (e.g., APIs)|

```yaml

# LoadBalancer Example

apiVersion: v1

kind: Service

metadata:

name: nginx-service

spec:

type: LoadBalancer

ports:

- port: 80

selector:

app: nginx

```

---

### \*\*Q8: NodePort vs. LoadBalancer?\*\*

- \*\*NodePort:\*\*

- Exposes on worker node IPs (`<NodeIP>:30007`).

- Accessible within \*\*VPC/organization\*\*.

- \*\*LoadBalancer:\*\*

- Creates cloud LB (e.g., AWS ELB).

- Accessible \*\*publicly\*\* (internet).

---

### \*\*Q9: Role of Kubelet?\*\*

- Runs on \*\*worker nodes\*\*.

- Ensures Pods are healthy (restarts if crashed).

- Reports Pod status to API Server.

📌 \*\*Key Task:\*\* Monitors Pods → Triggers auto-healing via ReplicaSet.

---

### \*\*Q10: Day-to-Day Kubernetes Tasks?\*\*

1. \*\*Cluster Maintenance:\*\*

- Upgrading nodes, security patches.

2. \*\*Deployments:\*\*

- Rolling updates, rollbacks (`kubectl rollout`).

3. \*\*Troubleshooting:\*\*

- Debugging Pods (`kubectl describe/logs`).

4. \*\*Monitoring:\*\*

- Setting up Prometheus/Grafana.

5. \*\*RBAC/Networking:\*\*

- Managing access, network policies.

---

## \*\*2. Pro Tips for Interviews\*\*

- \*\*Practice Commands:\*\*

```sh

kubectl get pods -o wide

kubectl describe svc <name>

kubectl logs <pod-name>

```

- \*\*Understand YAML:\*\* Know `deployment.yaml`, `service.yaml` structures.

- \*\*Real-World Examples:\*\* Relate answers to projects (e.g., "In my last role, I used Deployments for zero-downtime updates").

---

## \*\*3. Summary\*\*

✅ Mastered \*\*10 key Kubernetes interview questions\*\*.

✅ Learned architecture, Services, Deployments, and day-to-day tasks.

✅ Prepared for real-world scenarios (troubleshooting, scaling).

🚀 \*\*Next:\*\* Kubernetes Ingress & Helm!

📢 \*\*Feedback?\*\* Comment below! 👍 \*\*Like & Share\*\* if this helped!

---

\*\*End of Notes\*\* 🎉