Got it 🚀 Let’s build a **Docker Interview Q\&A Guide** (just like we did for Kubernetes).
This will cover **basic → advanced → real-world scenario-based questions** so you can answer confidently in interviews.

---

# 🐳 Docker Most Asked Interview Questions & Answers

---

## **Basics**

### 1. **What is Docker?**

* Docker is a **containerization platform** that allows applications to run in **isolated environments** called **containers**.
* Unlike VMs, containers share the host OS kernel, making them **lightweight & faster**.

---

### 2. **What is the difference between a container and a VM?**

* **VM**: Runs on a hypervisor, includes a full guest OS → heavier.
* **Container**: Runs on Docker engine, shares host OS → lightweight, faster startup.

---

### 3. **What is a Docker Image?**

* A **read-only template** used to create containers.
* Built from instructions in a `Dockerfile`.
* Example: `nginx:latest` is an image from Docker Hub.

---

### 4. **What is a Docker Container?**

* A **running instance of a Docker image**.
* Example:

  ```bash
  docker run -d -p 80:80 nginx
  ```

---

### 5. **What is the difference between Docker `COPY` and `ADD` in Dockerfile?**

* **COPY** → copies files/folders into image (preferred).
* **ADD** → same as COPY but also supports remote URLs & auto-extraction of tar files.

---

---

## **Intermediate**

### 6. **How do you check running containers?**

```bash
```

```
docker ps
docker ps -a   # includes stopped containers
```

---

### 7. **How do you persist data in Docker?**

* By using **Volumes** (managed by Docker) or **Bind mounts** (host directory).
* Example:

  ```bash
  docker run -d -v mydata:/var/lib/mysql mysql
  ```

---

### 8. **What is the difference between `docker run` and `docker exec`?**

* `docker run` → starts a new container.
* `docker exec` → runs a command inside an existing container.

---

### 9. **What is the difference between CMD and ENTRYPOINT in Dockerfile?**

* **CMD** → default command, can be overridden.
* **ENTRYPOINT** → fixed command, harder to override.
* Example:

  ```dockerfile
  ENTRYPOINT ["python"]
  CMD ["app.py"]
  ```

  → Always runs `python app.py`.

---

### 10. **How do you reduce the size of a Docker image?**

* Use smaller base images (e.g., `alpine`).
* Combine RUN commands → reduce layers.
* Use `.dockerignore` to exclude unnecessary files.

---

---

## **Advanced**

### 11. **How do you check logs of a container?**

```bash
docker logs <container_id>
docker logs -f <container_id>   # follow logs
```

---

### 12. **How do you connect multiple containers?**

* Use **Docker Networks**.
* Example:

  ```bash
  docker network create mynet
  docker run -d --network=mynet --name=db mysql
  docker run -d --network=mynet --name=app myapp
  ```

---

### 13. **What is the difference between Docker Compose and Docker Swarm?**

* **Docker Compose** → defines multi-container apps (`docker-compose.yml`).
* **Docker Swarm** → container orchestration tool (clusters, scaling, HA).

---

### 14. **What is the difference between Dockerfile and docker-compose.yml?**

* **Dockerfile** → defines how to build an image.
* **docker-compose.yml** → defines how to run multiple services together.

---

### 15. **How do you secure Docker containers?**

* Use minimal base images.
* Run containers as non-root.
* Use Docker Bench for security.
* Apply resource limits (`--memory`, `--cpus`).

---

---

## **Scenario-Based**

### 16. **A container keeps restarting. How do you debug?**

* Check logs: `docker logs <container>`.
* Check container status: `docker ps -a`.
* Run interactively: `docker run -it <image> /bin/bash`.
* Fix misconfigured command/env.

---

### 17. **Your container cannot connect to the internet. What do you check?**

* Verify Docker network driver (`bridge`, `host`).
* Check firewall rules (iptables).
* Restart Docker service.

---

### 18. **How do you run a container in the background?**

```bash
docker run -d nginx
```

---

### 19. **How do you update a running container with a new image version?**

```bash
```

```
docker pull nginx:latest
docker stop mynginx
docker rm mynginx
docker run -d --name=mynginx -p 80:80 nginx:latest
```

---

### 20. **How do you monitor Docker containers?**

* Commands:

  ```bash
  docker stats
  docker inspect <container>
  ```
* Tools: Prometheus, cAdvisor, Grafana.

---

## **Docker in Real-World**

* Companies usually use Docker with **Kubernetes** for orchestration.
* You may also be asked:

  * How to store images? → **Private Registry (ECR, GCR, ACR, Harbor)**
  * CI/CD with Docker → Build, push, deploy workflow.

---

✅ With these **20 Q\&A (basic → advanced → scenarios)** you'll be well-prepared for Docker interviews.

---

👉 Do you want me to also prepare a **Docker + Kubernetes combined interview question set** (since many companies ask them together)?

---

Perfect ✅ Most companies **don't ask Docker and Kubernetes separately** — they want to know if you can use **Docker for containerization** and **Kubernetes for orchestration**.

Here's a **combined Docker + Kubernetes Interview Q\&A Guide** with **basics → intermediate → advanced → real-world scenarios**.

---

# 🐳⚓ Docker + Kubernetes Interview Questions & Answers

---

## **1. Basics**

### Q1. What is the difference between Docker and Kubernetes?

* **Docker** → Containerization platform (build & run containers).
* **Kubernetes (K8s)** → Orchestration platform (manage containers at scale).

Example:

* Docker = A shipping container.

* Kubernetes = The port managing 1000s of containers.

---

### Q2. Can Kubernetes run without Docker?

* Yes ✅ Kubernetes does not depend only on Docker.
* It supports other container runtimes (CRI) like **containerd, CRI-O, Podman**.

---

### Q3. What is the difference between a Pod and a Container?

* **Container** → Smallest unit of execution (runs app).
* **Pod** → Abstraction in Kubernetes that can hold **one or multiple containers** sharing the same network/storage.

---

---

## **2. Intermediate**

### Q4. How do you scale an application in Docker vs Kubernetes?

* **Docker**:

  ```bash
  docker run -d --name web1 nginx
  docker run -d --name web2 nginx
  ```

  (manual scaling).

* **Kubernetes**:

  ```bash
  kubectl scale deployment nginx-deploy --replicas=5
  ```

  (automatic scaling using HPA).

---

### Q5. How do you expose applications in Docker vs Kubernetes?

* **Docker**: `docker run -p 8080:80 nginx`
* **Kubernetes**:

  ```yaml
  kind: Service
  apiVersion: v1
  metadata:
    name: my-service
  spec:
    type: NodePort
    ports:
      - port: 80
        targetPort: 80
        nodePort: 30080
    selector:
      app: nginx
  ```

---

### Q6. How do you persist data in Docker and Kubernetes?

* **Docker**: Volumes → `docker run -v mydata:/var/lib/mysql mysql`
* **Kubernetes**: PersistentVolume (PV) + PersistentVolumeClaim (PVC).

---

### Q7. What is the difference between Docker Compose and Kubernetes?

* **Docker Compose** → Define & run multi-container apps on a single host.
* **Kubernetes** → Run multi-container apps across a **cluster** (scalable, fault-tolerant).

---

---

## **3. Advanced**

### Q8. How do you troubleshoot a container not starting?

* **Docker**:

  ```bash
  docker logs <container_id>
  docker inspect <container_id>
  ```
* **Kubernetes**:

  ```bash
  kubectl describe pod <pod_name>
  kubectl logs <pod_name> -c <container>
  kubectl get events --sort-by=.metadata.creationTimestamp
  ```

---

### Q9. What is the difference between Docker Swarm and Kubernetes?

* **Swarm** → Native Docker clustering, easier but less feature-rich.
* **Kubernetes** → Industry standard, powerful (self-healing, autoscaling, RBAC, secrets).

---

### Q10. How do you update an application in Docker vs Kubernetes?

* **Docker**: Stop container → Pull new image → Run again.
* **Kubernetes**: Rolling updates with Deployments.

  ```bash
  kubectl set image deployment/nginx-deploy nginx=nginx:latest
  ```

---

### Q11. How do you monitor containers in Docker vs Kubernetes?

* **Docker**:

  ```bash
  docker stats
```

  docker inspect
  ```
* **Kubernetes**: Metrics Server, Prometheus + Grafana, kubectl top.

---

---

## **4. Real-World Scenarios**

### Q12. Your containerized app works in Docker but fails in Kubernetes. What could be wrong?

* Missing environment variables or ConfigMaps.
* Pod can’t reach other services → check Services & NetworkPolicies.
* Persistent volume not bound → PVC mismatch.

---

### Q13. How do you deploy a microservices app using Docker and Kubernetes?

* **Step 1**: Package each service into a Docker image (`Dockerfile`).
* **Step 2**: Push images to a registry (ECR, ACR, Docker Hub).
* **Step 3**: Write Kubernetes YAML (Deployment, Service, Ingress).
* **Step 4**: Apply configs:

  ```bash
  kubectl apply -f k8s/
  ```

---

### Q14. How do you handle secrets in Docker and Kubernetes?

* **Docker**: Pass via `--env` or `--env-file` (less secure).
* **Kubernetes**: Use `Secret` object. Example:

  ```yaml
  apiVersion: v1
  kind: Secret
  metadata:
    name: db-secret
  type: Opaque
  data:
    username: YWRtaW4=
    password: cGFzc3dvcmQ=
  ```

---

### Q15. What happens if a node crashes in Kubernetes vs Docker?

* **Docker**: Containers on that node stop. Manual restart required.
* **Kubernetes**: Pods are automatically rescheduled on healthy nodes.

---

---

## **5. Common Interview Flow**

👉 Most interviews follow this **progression**:

1. **Docker Basics** → Images, Containers, Volumes.
  ```

2. **Docker Scenarios** → Debugging, Scaling, Networking.
3. **Kubernetes Basics** → Pods, Deployments, Services.
4. **Kubernetes Scenarios** → Debugging Pods, Autoscaling, Networking.
5. **Docker + K8s Real-World** → CI/CD pipeline, monitoring, security.

---

🔥 With this guide, you can **connect Docker + Kubernetes answers** instead of answering separately.

---

---

Perfect 👍 — Interviewers **love scenario-based Docker questions** because they want to see if you can debug and solve **real problems in production**.

Here’s a list of **most asked scenario-based Docker interview questions + how to answer them** 🚀:

---

# 🐳 Scenario-Based Docker Interview Questions

---

## **1. Container Fails to Start**

**Q:** Your Docker container keeps exiting immediately after running. How will you debug?
**A:**

* Check logs:

  ```bash
  docker logs <container_id>
  ```
* Inspect container config:

  ```bash
  docker inspect <container_id>
  ```
* Run interactively for debugging:

  ```bash
  docker run -it <image> /bin/bash
  ```

👉 Common reasons: Wrong entrypoint, app crash, missing dependencies.

---

## **2. Port Not Accessible**

**Q:** You run `docker run -d -p 8080:80 nginx`, but the app is not accessible. What will you check?
**A:**

1. Is container running? → `docker ps`
2. Is service listening inside container? → `docker exec -it <id> netstat -tulnp`
3. Is firewall blocking port?
4. Did you map the port correctly (`-p hostPort:containerPort`)?

---

## **3. Data Lost After Restart**

**Q:** You ran MySQL inside Docker, but after restarting the container, all data is lost. Why?
**A:**

* Data was stored inside container filesystem → deleted when container stopped.
* Fix → Use **volumes**:

  ```bash
  docker run -v mydb:/var/lib/mysql mysql
  ```

---

## **4. Two Containers Need to Talk**

**Q:** You have an app container and a database container. The app cannot connect to DB. What will you check?
**A:**

* Are they on the same network?

  ```bash
  docker network ls
  docker network connect <network> <container>
  ```
* Use container name instead of `localhost`.
  👉 Example with Docker Compose (networking is automatic):

```yaml
services:
  db:
    image: mysql
  app:
    image: myapp
    depends_on:
      - db
```

---

## **5. High CPU/Memory Usage**

**Q:** Your container is consuming too much memory and killing the host machine. How will you fix?
**A:**

* Monitor:

  ```bash
  docker stats
  ```
* Apply limits:

  ```bash
  docker run -m 512m --cpus=1 nginx
  ```
* Use resource requests/limits in orchestration (Kubernetes/Swarm).

---

## **6. Image Size Too Large**

**Q:** Your Docker image is **2GB**. How will you reduce it?
**A:**

* Use smaller base image (e.g., `alpine` instead of `ubuntu`).
* Combine RUN statements to reduce layers.
* Remove unnecessary files with `.dockerignore`.
* Multi-stage builds:

  ```dockerfile
  FROM golang:1.20 AS builder
  WORKDIR /app
  COPY . .
  RUN go build -o main .

  FROM alpine:latest
  COPY --from=builder /app/main /main
  CMD ["/main"]
  ```

---

## **7. Stale Containers/Images Taking Space**

**Q:** Your server is running out of disk space due to old Docker images and containers. How will you clean up?
**A:**

```bash
docker system df
docker system prune -a
docker volume prune
```

---

## **8. Application Works Locally but Fails in Container**

**Q:** App runs fine on your host, but fails in Docker. How will you debug?
**A:**

* Compare host vs container environment.
* Check missing dependencies inside container.
* Validate environment variables (`docker inspect`).
* Debug with:

  ```bash
  docker exec -it <container> /bin/bash
  ```

---

## **9. Container Crashes on Restart**

**Q:** Your container keeps restarting with policy `--restart always`. How do you debug?
**A:**

* Check logs: `docker logs -f <id>`
* Inspect exit code: `docker inspect --format='{{.State.ExitCode}}' <id>`
* Fix underlying issue (bad config, DB not reachable, permission denied).

---

## **10. You Need Zero-Downtime Deployment**

**Q:** How will you update a Dockerized app with zero downtime?
**A:**

* Use **reverse proxy + multiple containers**:

  * Run new version (`v2`) in a new container.
  * Update load balancer (NGINX, HAProxy) to route traffic.
  * Stop old container after verifying.
* Or use orchestration tool (Kubernetes rolling update).

---

✅ These scenarios test:

* **Troubleshooting skills** (logs, inspect, exec).
* **Best practices** (volumes, networks, resource limits).
* **Real-world ops knowledge** (cleanup, monitoring, scaling).

---

👉 Do you want me to also create a **mock Q\&A flow** where I play interviewer and ask you scenario-based Docker questions, and you try answering? That way you'll **practice live interview situations**.