Mock Interview Guide Docker

🗃️ **Instructions for Interviewer:**

- You are playing the role of **interviewer**. Use this guide as a script.
- Ask each question one at a time. Follow the steps: **Definition → Details → Scenario → Follow-up.**
- If the interviewee struggles, use the **hint**.
- The goal is to keep it conversational and practical. Help the interviewee think and express their learning.
- **colors assigned:** **Questions** **Answers** **Hint**

Freshers - Level

Docker (10 Easy DevOps Interview Questions)

1. 🗣️ **"Can you explain what Docker is?"**

   ✅ Expected Answer: "Docker is a platform used to create and run containers. Containers are lightweight and portable environments that package an app with all its dependencies, so it works the same on any system."

   💡 Hint (if needed): "Think of Docker like a sealed box that includes your app, OS libraries, and system tools. It solves the 'it works on my machine' problem."

2. 🗣️ **"What is the difference between Docker containers and virtual machines?"**

   ✅ Expected Answer: "Containers share the host OS and are lightweight, while VMs run full OS instances and are heavier. Docker starts faster and uses fewer resources."

💡 Hint: “Think about the OS layer. Do both Docker and VMs run their own OS?”

3. 🗣️ **“What is the difference between a Docker image and a container?”**

✅ **Expected Answer: “An image is a read-only template with app and settings. A container is the running instance of that image.”**

💡 **Hint: “Image is like a recipe; container is the final dish.”**

4. 🗣️ **“What is a Dockerfile?”**

✅ **Expected Answer: “A Dockerfile is a script with instructions to build a Docker image. It has commands like FROM, RUN, COPY, CMD.”**

💡 **Hint: “It’s like a list of steps to prepare your Docker image.”**

5. 🗣️ **“How do you create a Docker image from a Dockerfile?”**

✅ **Expected Answer: “Use the command: docker build -t image-name .”**

💡 **Hint: “Do you remember the docker build command?”**

6. 🗣️ **“How do you run a Docker container from an image?”**

✅ **Expected Answer: “Use the command: docker run image-name You can add -d to run in background or -p to map ports.”**

💡 **Hint: “Have you seen the docker run command?”**

7. 🗣️ **“How do you list all running containers?”**

✅ **Expected Answer: “Use the command: docker ps For all containers including stopped: docker ps -a”**

💡 **Hint: “It’s similar to the ps command in Linux.”**

8. 🗣️ **“How do you stop and remove a Docker container?”**

✅ **Expected Answer: “Use: docker stop container-id Then: docker rm container-id”**

💡 **Hint: “Two steps: first stop, then remove.”**

9. 🗣️ **“What are Docker volumes used for?”**

✅ **Expected Answer: “Volumes are used to persist data outside containers. So data remains even if the container is deleted.”**

💡 **Hint: “Think about saving data from a database container.”**

10. 🗣️ **“What is Docker Compose used for?”**

✅ Expected Answer: “Docker Compose is used to run multi-container applications using a YAML file (docker-compose.yml).”

💡 Hint: “Have you used a YAML file to define services like web + db together?”

SCENARIO-BASED QUESTIONS

1. Ask:🗣️ “You created a Docker container but it's not accessible from the browser. What could be wrong?”

✔️ Expected: “You probably forgot to expose or publish the port using -p.”

💡 Hint: “Did you use -p 8080:80 or EXPOSE in the Dockerfile?”

2. Ask:🗣️ “Your container stops immediately after running. What might be the issue?”

✔️ Expected: “The main process may be exiting. Docker containers stop when the main command ends.”

💡 Hint: “What command did you specify in CMD or ENTRYPOINT?”

3. Ask:🗣️ “You installed something in a container, but it's gone when you restart. Why?”

✔️ Expected: “Changes made inside a running container are not persistent unless committed or built into a Dockerfile.”

💡 Hint: “Was it saved into a new image or just modified at runtime?”

4. Ask:🗣️ "You need to share logs or data between your host and container. What will you use?"

> **✔️ Expected: "Use volumes or bind mounts using -v or --mount."**
>
> **💡 Hint: "Data persistence? Think Docker volumes."**

PROJECT-BASED QUESTIONS

5. Ask :🗣️ "You want to containerize a basic Python app. How would you start?"

> **✔️ Expected: "Write a Dockerfile with Python base image, copy app code, install requirements, then build and run."**
>
> **💡 Hint: "Dockerfile → Build → Run with ports."**

6. Ask: 🗣️ "Your friend wants to run your app without installing anything. What can you give them?"

> **✔️ Expected: "A Docker image via Docker Hub or a docker-compose.yml file to replicate my environment."**
>
> **💡 Hint: "Docker Hub or Docker Compose?"**

Medium-Level

Docker (DevOps Interview Questions - 1 to 2 Years Experience)

1. Ask: 🗣️ **"How does Docker differ from traditional deployment methods?"**

   ✅ **Expected Answer: "Traditional methods involve manual setup and dependency installation. Docker provides consistent environments using containers, making deployment faster and more reliable."**

   💡 **Hint: "Think about environment consistency and speed of deployment."**

2. Ask: 🗣️ **"What is the role of the Docker daemon?"**

   ✅ **Expected Answer: "Docker daemon (dockerd) manages Docker containers, images, networks, and storage volumes. It listens for Docker API requests."**

   💡 **Hint: "It's the background service that powers all Docker commands."**

3. Ask: 🗣️ **"Can you explain what happens when you run docker build?"**

   ✅ **Expected Answer: "Docker reads the Dockerfile instructions and creates an image layer-by-layer from top to bottom."**

   💡 **Hint: "Think of each Dockerfile instruction creating a new image layer."**

4. Ask: 🗣️ **"What is the difference between ENTRYPOINT and CMD in a Dockerfile?"**

✅ **Expected Answer: "CMD provides default arguments. ENTRYPOINT defines the main command. If both are used, CMD supplies arguments to ENTRYPOINT."**

💡 **Hint: "ENTRYPOINT is fixed; CMD is flexible."**

5. Ask: 🗣️ **"How can you persist data in Docker containers?"**

   ✅ **Expected Answer: "By using Docker volumes or bind mounts. Volumes are managed by Docker; bind mounts map to host directories."**

   💡 **Hint: "Think of saving logs or DB data that shouldn't disappear when container is deleted."**

6. Ask: 🗣️ **"How does Docker networking work?"**

   ✅ **Expected Answer: "Docker creates a default bridge network. Containers in the same network can communicate. You can also create custom networks."**

   💡 **Hint: "Have you tried pinging containers inside a user-defined bridge network?"**

7. Ask: 🗣️ **"What is the use of .dockerignore file?"**

   ✅ **Expected Answer: "It excludes specified files/folders from the build context. This speeds up builds and avoids sending unnecessary files."**

   💡 **Hint: "Similar to .gitignore – helps avoid copying junk into your image."**

8. Ask: 🗣️ **"What is the difference between docker stop and docker kill?"**

✅ **Expected Answer: "docker stop sends SIGTERM for graceful shutdown; docker kill sends SIGKILL to force stop."**

💡 **Hint: "One is gentle, the other is instant termination."**

9. Ask: 🗣️ **"How do you troubleshoot a container that keeps restarting?"**

   ✅ **Expected Answer: "Check logs using docker logs. Use docker inspect to examine settings. Also check Dockerfile and application errors."**

   💡 **Hint: "Try to understand the container's startup process using logs and inspect."**

10. Ask: 🗣️ **"What is a multi-stage build in Docker and why is it useful?"**

    ✅ **Expected Answer: "Multi-stage builds allow using one image for building and another for running, reducing final image size."**

    💡 **Hint: "Think about separating build tools and runtime – only keep what's needed to run."**

SCENARIO-BASED QUESTIONS

7. Ask: 🗣️ "You want to run a database container and ensure data isn't lost after restart. How would you do it?"

✔️ **Expected: "Mount a named volume to the DB's data directory, like -v pgdata:/var/lib/postgresql/data."**

💡 **Hint: "What keeps data between restarts?"**

8. Ask: 🗣️ "You built a new image but it's 2GB in size. How can you reduce it?"

> **✔️ Expected: "Use multi-stage builds, lighter base images (like alpine), and clean up temp files."**
>
> **💡 Hint: "What's the base image and layers doing?"**

9. Ask: 🗣️ "You have 3 containers that need to communicate internally. What's the best approach?"

> **✔️ Expected: "Use a user-defined bridge network or Docker Compose — they can talk by container name."**
>
> **💡 Hint: "Default bridge doesn't do DNS. Try custom network."**

10. Ask: 🗣️ "You restarted Docker and lost your containers. What happened?"

> **✔️ Expected: "They weren't run with --restart=always or saved in Compose/Kubernetes."**
>
> **💡 Hint: "Docker doesn't keep them by default after daemon restart."**

PROJECT-BASED QUESTIONS

11. Ask: 🗣️ "You're building a CI/CD pipeline with Dockerized services. How do you test and deploy?"

✔️ Expected: “Use Docker Compose to spin up services for integration tests, then push built image to Docker Hub.”

💡 Hint: “Think local test → build → push → deploy.”

12. Ask: 🗣️ “You want to deploy your Node.js + Redis app in dev and prod using Docker. How would you structure it?”

✔️ Expected: “Create a docker-compose.yml with multiple services, use environment variables, volumes for dev, and scale in prod.”

💡 Hint: “Compose + .env + service scaling.”

Advanced-Level

(DevOps Interview Questions - 3+ Years Experience)

1. Ask: 🗣️ “Can you explain how Docker handles image layering and how it impacts performance and storage?”

   ✅ Expected Answer: Docker builds images in layers based on Dockerfile instructions. Layers are cached and reused across builds, which speeds up image creation and saves storage. Modifying a layer invalidates all layers after it.

   💡 Hint: “Why do we try to keep frequently changing lines at the bottom of a Dockerfile?”

2. Ask: 🗣️ “How do you reduce the size of a Docker image in production environments?”

   ✅ Expected Answer: Use smaller base images (like alpine),

remove unnecessary packages, use multi-stage builds, and clear caches (apt-get clean, pip cache purge).

💡 Hint: "Have you used multi-stage builds or stripped down images for production?"

3. **Ask: 🗣️ "What are the implications of running a container in privileged mode?"**

   ✅ Expected Answer: It gives the container access to all host devices and allows more low-level operations. It's insecure and should be avoided unless absolutely necessary.

   💡 Hint: "Think about how this could be used to access host resources dangerously."

4. Ask: 🗣️ **"Can you explain how Docker content trust works?"**

   ✅ Expected Answer: Docker Content Trust (DCT) uses digital signatures to verify image authenticity. It ensures the image is from a trusted source.

   💡 Hint: "Have you used DOCKER_CONTENT_TRUST=1 to enable this?"

5. Ask: 🗣️ **"How do you secure Docker containers in a production environment?"**

   ✅ Expected Answer:

   - **Use non-root users inside containers**
   - **Limit container capabilities (with --cap-drop)**
   - **Scan images for vulnerabilities**

- **Use private registries and signed images**
- **Restrict networking and volumes**

💡 **Hint: "Think about both build-time and run-time hardening."**

6. **Ask: 🗣️ "What is the difference between Docker Swarm and Kubernetes?"**

   ✅ **Expected Answer: Docker Swarm is a simpler orchestrator tightly integrated with Docker. Kubernetes is more complex, flexible, and widely used for large-scale deployments.**

   💡 **Hint: "Which one did you use and why? What challenges did you face?"**

7. **Ask: 🗣️ "How do you monitor Docker containers in a real-time production setup?"**

   ✅ **Expected Answer: Use tools like Prometheus, Grafana, ELK stack, or Docker stats. Metrics like CPU, memory, network IO, and container logs are key.**

   💡 **Hint: "Did you expose Docker metrics using cadvisor or integrate with monitoring tools?"**

8. Ask: 🗣️ **"Can you explain how image vulnerability scanning works in Docker?"**

   ✅ **Expected Answer: Tools like Trivy, Clair, or Docker Hub scan image layers for known CVEs. These tools compare installed packages against vulnerability databases.**

   💡 **Hint: "Have you set up automated scans or integrated them into CI/CD?"**

9. Ask: 🗣️ **“How do you handle secret management in containers?”**

✅ **Expected Answer: Avoid hardcoding secrets in images or ENV variables. Use tools like Docker secrets (in Swarm), HashiCorp Vault, AWS Secrets Manager, or Kubernetes secrets.**

💡 **Hint: “Where do you store DB passwords, API keys, etc.?”**

10. Ask: 🗣️ **“What would you do if a containerized app suddenly starts consuming high CPU and memory?”**

✅ **Expected Answer:**

- **Use docker stats to inspect container resource usage**
- **Check application logs**
- **docker top to see running processes**
- **docker inspect for config issues**
- **Restart with resource limits**

💡 **Hint: “What would your investigation steps look like? How would you prevent this in future?”**

SCENARIO-BASED QUESTIONS

13. Ask: 🗣️ “You notice high CPU usage in one container. How do you isolate and limit its resources?”

✔️ Expected: “Use Docker stats to monitor, and limit resources via -cpus and --memory flags.”

💡 Hint: “Try resource constraints at runtime.”

14. Ask: 🗣️ “You need to scan your Docker images for security vulnerabilities. What tool would you use?”

✔️ Expected: “Tools like Trivy, Docker scan, or integrate scanning in CI pipeline.”

💡 Hint: “Image scanning before deploy?”

15. Ask: 🗣️ “You’re building images in CI and want faster builds. Any optimization tips?”

✔️ Expected: “Use build cache, .dockerignore, multi-stage builds, and avoid ADD-ing large directories.”

💡 Hint: “Smaller context = faster build.”

16. Ask:🗣️ “You’ve deployed a container that’s failing, but logs aren’t helping. How else can you debug?”

✔️ Expected: “Use docker exec to inspect live container, attach shell, or use --entrypoint bash to manually run.”

💡 Hint: “Get inside the container.”

PROJECT-BASED QUESTIONS

17. Ask:🗣️ “You're designing a Docker-based microservice system. How would you handle service discovery and networking?”

✔️ Expected: “Use Docker Compose for local, or orchestrators (K8s, Swarm) with DNS and overlay networks.”

💡 Hint: “Container-to-container name resolution.”

18. Ask:🗣️ “You're asked to set up a local dev environment with 5+ interdependent services. How would you handle this?”

✔️ Expected: “Use Docker Compose with separate Dockerfiles, shared networks, and mount source for hot reload.”

💡 Hint: “Multi-container setup + config mgmt.”