difference between docker exec and docker run

**ChatGPT said:**

Great question 👍 — this is a common point of confusion.  
Here’s the **difference between docker run and docker exec:**

**🔹 docker run**

* **Creates and starts a new container** from an image.
* If the image doesn’t exist locally, Docker pulls it from a registry.
* Each docker run starts a **fresh container** with its own filesystem, environment, and process.

**Example:**

docker run -it ubuntu bash

* Pulls the Ubuntu image (if not present).
* Creates a **new container** from it.
* Starts an interactive bash shell inside it.

👉 Every docker run = **new container**.

**🔹 docker exec**

* **Runs a command in an already running container.**
* Does not create a new container, it just attaches to an existing one.
* Useful for debugging, checking logs, or running extra commands inside the container.

**Example:**

docker exec -it mycontainer bash

* Runs bash inside the **existing container** named mycontainer.
* Container must already be running, otherwise it fails.

👉 Every docker exec = **extra process in an existing container**.

**🔹 Key Differences Table**

| **Feature** | **docker run** | **docker exec** |
| --- | --- | --- |
| Purpose | Start a **new container** | Run a command in an **existing container** |
| Container lifecycle | Creates a new one each time | Uses an already running container |
| Image dependency | Needs an image to start | Needs a container ID/name |
| Typical use | Starting apps/services | Debugging, inspecting, or running extra commands |

✅ Example workflow:

1. Start a container:
2. docker run -d --name webapp nginx

(Runs NGINX in a new container named webapp.)

1. Exec into it:
2. docker exec -it webapp bash

(Opens a bash shell inside the **already running** NGINX container.)

# 🐳 Top 20 Docker Interview Q&A

### ****1. What is Docker and why is it used?****

👉 Docker is a containerization platform that packages applications with dependencies into portable containers.  
**Why?** → Consistency, faster deployments, lightweight, microservices support.

### ****2. Difference between a Docker image and container?****

👉 **Image** = blueprint (read-only).  
👉 **Container** = running instance (read-write).

### ****3. How does Docker achieve isolation?****

👉 Uses Linux **namespaces** (process, network, PID) + **cgroups** (resource limits) + **UnionFS** (layered filesystem).

### ****4. What happens when you run**** docker run nginx****?****

1. Docker daemon checks if nginx image exists locally.
2. If not, pulls from registry.
3. Creates a container from image.
4. Assigns network, mounts volumes.
5. Runs default command (CMD).

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

👉 docker logs <container\_id>  
👉 For real-time: docker logs -f <id>

### ****6. Difference between**** docker run ****and**** docker exec****?****

👉 docker run → Starts a new container.  
👉 docker exec → Executes a command in an **existing container**.

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

👉 Use **volumes** (docker volume create) or bind mounts.  
👉 Example: docker run -v /host/data:/container/data.

### ****8. How do containers communicate with each other?****

👉 Via **Docker networks**:

* bridge (default, local machine).
* host (uses host’s network).
* overlay (multi-host, used in Swarm/K8s).

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

* Use **alpine base images**.
* Use **multi-stage builds**.
* Clean caches (apt-get clean && rm -rf /var/lib/apt/lists/\*).

### ****10. What is a Dockerfile?****

👉 A text file with instructions to build a Docker image.  
Example:

FROM python:3.9-slim

COPY app.py /app/

CMD ["python", "/app/app.py"]

### ****11. How do you share Docker images?****

👉 Push to a registry:

* Public: Docker Hub.
* Private: AWS ECR, GCP GCR, Harbor.  
  Command:

docker tag myapp:v1 repo/myapp:v1

docker push repo/myapp:v1

### ****12. What is the difference between Docker and a VM?****

| **Feature** | **VM (Virtual Machine)** | **Docker (Container)** |
| --- | --- | --- |
| OS | Full OS per VM | Shares host OS kernel |
| Size | GBs | MBs |
| Startup | Minutes | Seconds |
| Isolation | Strong | Process-level |

### ****13. What is Docker Compose?****

👉 A tool to manage **multi-container apps** with YAML file.  
Example:

services:

web:

image: nginx

ports: ["8080:80"]

db:

image: mysql

Run → docker-compose up -d

### ****14. How do you debug a container stuck in**** CrashLoopBackOff****?****

* docker ps -a → check status.
* docker logs <id> → check logs.
* docker inspect <id> → check config/env.
* docker exec -it <id> bash → enter container.

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

* CMD → Default command, can be overridden at runtime.
* ENTRYPOINT → Defines fixed command, cannot be overridden easily.

### ****16. What is a multi-stage build in Docker?****

👉 Technique to build lean images by separating build and runtime.  
Example:

FROM golang:1.19 AS builder

WORKDIR /app

COPY . .

RUN go build -o app .

FROM alpine:latest

COPY --from=builder /app/app /app

CMD ["/app/app"]

Result → small production image.

### ****17. How do you limit resources for a container?****

👉 Example:

docker run -m 512m --cpus="1.5" nginx

* -m → memory.
* --cpus → CPU cores.

### ****18. What are Docker namespaces and cgroups?****

* **Namespaces** → Provide isolation (PID, NET, IPC, MNT, UTS).
* **Cgroups** → Limit and monitor resources (CPU, memory, disk, network).

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

* Run as non-root user.
* Use minimal images.
* Scan images for vulnerabilities (trivy, anchore).
* Use secrets manager (not ENV vars).
* Enable AppArmor/SELinux profiles.

### ****20. Real-world scenario:****

**Q: Your container keeps restarting in production. How do you debug?**  
👉 Steps:

1. docker ps -a → Check status.
2. docker logs <id> → Error logs.
3. docker inspect <id> → Config/env/ports.
4. Check resources (CPU/mem).
5. If app-level bug → fix code & rebuild.
6. If infra-level → check ports, networking, volumes.