

Search...

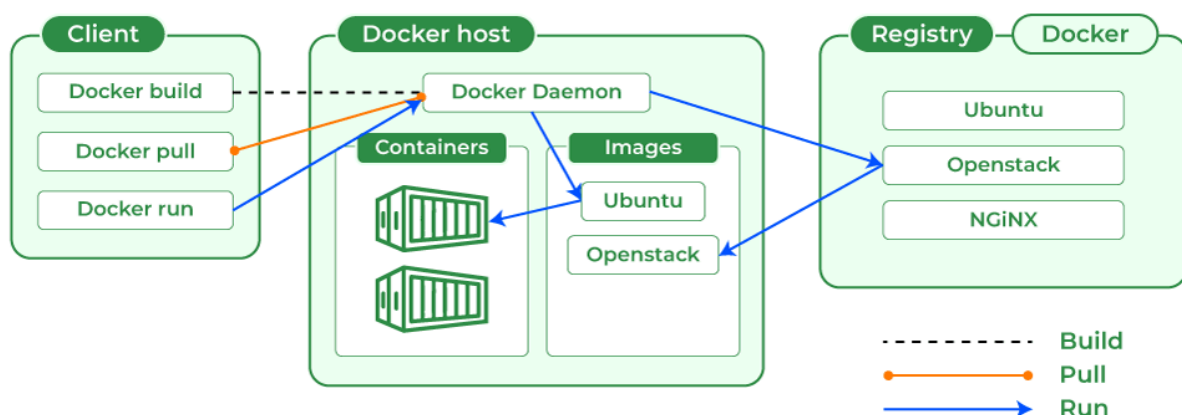
[DevOps Lifecycle](#) [DevOps Roadmap](#) [Docker Tutorial](#) [Kubernetes Tutorials](#) [Amazon Web Services \[AWS\]](#)

# Top 50+ Docker Interview Questions and Answers (2024)

Last Updated : 16 Oct, 2024

Docker is an open-source platform that simplifies the deployment, scaling, and management of applications using lightweight containers. It has transformed the way apps are built, shipped, and deployed, becoming a key tool for many top companies like **Uber**, **Airbnb**, **Google**, **Netflix**, and **Amazon**. Docker is popular for its efficiency, scalability, and ease of use, making it a preferred choice in modern DevOps.

In this interview preparation guide, we've put together over 50 Docker interview questions and answers. These cover everything from basic Docker containers to advanced topics like **container orchestration**, **Docker Compose**, and **networking**. Now, whether you are just starting with Docker or have years of experience, this guide will help you prepare for a Docker-focused interview.



## Table of Content

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

Got It !

- [Scenario-Based Docker Interview Questions and Answers](#)

## What is Docker Container

**Docker containers** are lightweight a single server or ( [Virtual Machine](#) ) VM can run several containers simultaneously. It implements a high-level API to provide these lightweight containers that run processes in isolation. The magic of docker lies in making the process as containers ( [Operating system](#) ) by providing separate namespaces, security groups, and unique hostnames.

Docker comes up with popular tools such as docker-compose for defining and running, [docker swarm](#) providing the native functionality for docker [containers](#) and docker volume facilitating independent persistence of data.

## Basic Docker Interview Questions and Answers

### 1. What is Docker ?

[Docker](#) is a containerization platform that allows to package an application with all its dependencies into one single entity as single container which can be easily deployed and run on any machine that supports docker. This makes it easier to develop , test , deploy applications in different environments. It uses container technology to isolate processes and provide a lightweight, portable solution for application deployment.

### 2. What are the Features of Docker?

Docker features containerization for providing consistent deployment , using resources efficient shared kernel utilization, and provides seamless portability across environments. It enhances the security through isolation of containers supporting versioning and automated builds. It offers a rich number of pre-built images for streamlined application development and

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

## Pros of Docker

### Related searches

🔍 Devops Real Time Interview Questions

🔍 Devops Real Time Interview Questions

- **Portability:** It enables consistent deployment across various environments.
- **Resource Efficiency:** Optimizing of resource usage with a shared kernel will be done effectively.
- **Isolation:** It provides security through isolation of process and file system.
- **Automation:** it supports automated builds and streamlining development workflow

### Cons of Docker

- **Learning Curve:** Initial learning of the containerization concepts will be new to understand.
- **Additional Resources:** Containers use some more resources compared to running applications directly on host.
- **Security Concerns:** Misconfigurations may lead to the security risks if not properly managed.
- **Container Orchestration Complexity:** Management of orchestration tools will be complex for larger-scale deployments.

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

- **Docker Engine:** Docker engine is the runtime that executes containers.
- **Docker Images:** [Docker images](#) are lightweight, readable templates containing executable packages that include the application with its dependencies.
- **Docker Containers:** [Docker containers](#) are standardized , encapsulated environments that run applications/instances of Docker images.
- **Docker Compose:** [Docker compose](#) is a tool for defining and running multi-containered Docker applications.

## 5. Name and Explain the State of a Docker Container.

A state of a docker container directly influences its runtime characteristics and how it interacts with the underlying Operating system. A Docker container will be in one of these three states:

- A docker container will be in **running** state it is when actively executing.
- A container in **paused** state means that container is temporarily halted.
- The container will be in **stopped** state when it is inactive.

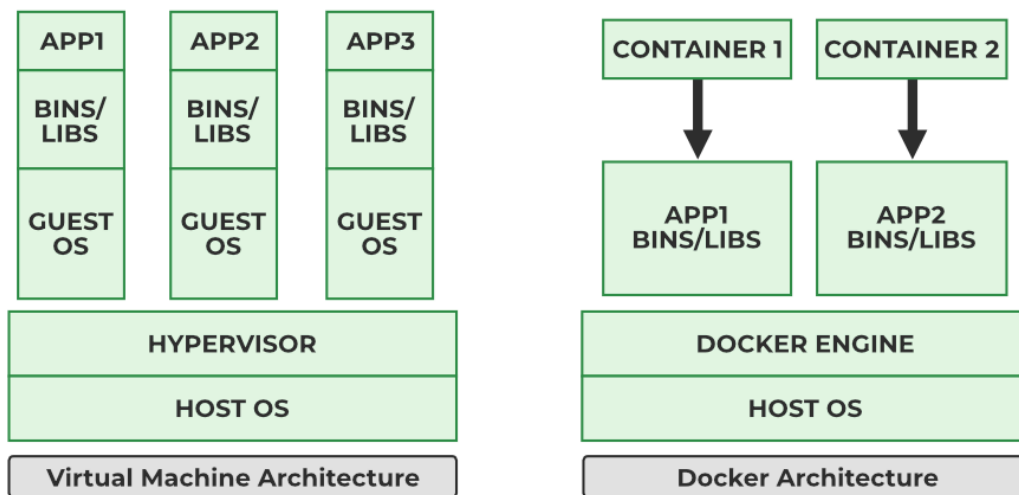
## 6. Can You tell What is the Functionality of a Hypervisor?

A [hypervisor](#) is a virtualization software that helps in running multiple operating systems (Guest OS) on a single physical host system by providing an isolation between the virtual machines (VMs) and manages their resources.

## 7. Difference between Docker and Virtualization?

Docker uses [containerization](#) concept, which shares the host OS kernel for efficiency and speed whereas Virtualization involves running complete OS instances ( Guest Operating systems ) on a hypervisor, which may have more overhead on using resources. The following figure illustrate on both

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).



## 8. On What Circumstances Will You Lose Data Stored in a Container?

The Data in a container can be lost whenever the container is deleted, or if docker non-[persistent storage](#) ( Ephemeral storage ) is used without proper data management. To make the data persistent , it is recommended to use Docker volumes or volume binding ( volume mounts ) are recommended.

## 9. What is Docker Hub?

[Docker Hub](#) is container registry that serves as a centralized repository for [Docker images](#). It built for developers and open source contributors to find , use , share and download container images. Docker Hub can be used either host public repos that can be used for free, or [docker private repos](#) for teams and enterprises.

## 10. What Command Can You Run to Export a Docker Image As an Archive?

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

```
docker save -o <output_file_name>.tar <image_name>
```

## 11. What Command Can Be Run to Import a Pre-Exported Docker Image Into Another Docker Host?

We can use this following command to import a pre-exported Docker image into another host:

```
docker load -i <input_file_name>.tar
```

## 12. Can a Paused Container Be Removed From Docker?

Yes, a paused container can be removed using the command with **rm** option:

```
docker rm <container_id>
```

## 13. How Do You get the Number Of Containers Running, Paused, and Stopped?

For obtaining the number of running, paused, and stopped containers in Docker you can use the command such as **`docker ps -q`** for knowing the list of running containers and **`docker ps -q -f "status=paused"`** for paused ones. Stopped containers can be counted using **`docker ps -aq -f "status=exited"`**. These commands will provide the list of container IDs, and you have to further process the output to get the counts programmatically like **`docker ps -q | wc -l`**.

The following command is used to know number of container are in running state:

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

The following command is used to know number of container are in paused state:

```
docker ps -aq -f "status=paused" | wc -l
```

The following command is used to know number of containers are in stopped state:

```
docker ps -aq -f "status=exited" | wc -l
```

## 14. How to Start, Stop, and Kill a Container?

In Docker to start , stop and kill a container we using start , stop and kill options on association with the docker command , the usage is given below.

To start the docker container use this command:

```
docker start < container_name >
```

To stop the docker container use this command:

```
docker stop < container_name >
```

To kill the docker container use this command:

```
docker kill < container_name >
```

## Intermediate Docker Interview Questions and Answers

### 15. Can You Tell the Difference Between CMD and ENTRYPOINT?

#### CMD vs ENTRYPOINT

- **CMD** is used for setting default commands and arguments that will be

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

- **ENTRYPOINT** configures a container to run as an executable, defining the command that has to be executed when the container starts. It is more strict than CMD and is oftenly used when the container should have to behave like an executable.

Know more about [CMD vs ENTRYPOINT](#).

## 16. What are the Key Distinctions Between Daemon Level Logging and Container Level Logging in Docker?

### Daemon Level Logging vs. Container Level Logging

- **Daemon Level Logging:** It involves with configuring the Docker daemon's logging behavior globally. This will affect all containers that are running on the host. Configuration settings will be applied at the daemon level.
- **Container Level Logging:** This will be connected to logs specific to an individual container. They can be accessed using the docker logs <container\_id> command , providing clear understanding regard the container's runtime activities.

```
docker logs <container_id>
```

## 17. What Does the Docker Info Command Do?

docker info provides detailed information regarding the Docker system. It includes information such as the number of containers, images, storage driver that are used and much more. It's a valuable command for gaining details on overview of the Docker environment.

## 18. Where are Docker Volumes Stored in Docker?

---

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).



## 19. Can You Tell the Differences Between a Docker Image and Layer?

A Docker Image can be considered as a snapshot of a file system and application dependencies. It is composed of multiple layers, where each layer will represent a set of filesystem changes. These Layers facilitate in efficient image creation and sharing common components among the images.

## 20. Can a Container Restart By Itself?

Yes, a container itself can restart automatically by setting up the `--restart` option during the creation period of time. For example using `'docker run --restart'` always. This will ensure that the container restarts irrespective of its exit status.

```
docker run --restart
```

## 21. Name the Essential Docker Commands and What They Do.

The essential Docker Commands are listed here:

- **docker run:** [Docker run command](#) is used to run the docker image as a docker container.
- **docker ps:** Docker os command will list all the running container in the docker.
- **docker exec:** It helps in execute the commands in a running container.
- **docker stop:** It will stops a container which is running in the docker.

## 22. What are Docker Object Labels?

Docker object labels are key-value mapping pair applied to the docker

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

```
docker run --label environment=production <image_name>
```

## 23. How Do You Check the Versions of Docker Client and Server?

Use `'docker version'` command to obtain the detailed information about the Docker client and server, including their respective version numbers.

```
docker version
```

## 24. Why is Docker System Prune Used? What Does It Do?

`'docker system prune'` is used for removal of unused data on inclusion of stopped containers, [docker networks](#), and [dangling images](#). It helps in freeing up the disk space on cleaning unnecessary resources.

```
docker system prune
```

## 25. What is Docker Swarm?

[Docker Swarm](#) is an inherited native clustering that comes up with a orchestration solution for the Docker software. It helps in simplifying the management of a swarm of Docker nodes on allowing the seamless scaling of the applications across various multiple nodes within the network. It provides built-in load balancing and will ensure the high availability of containerized applications.

## 26. How Do You Scale Docker Containers Horizontally?

Horizontal scaling is achieved through replicating the services across multiple nodes. Tools like [Docker Compose](#) or Docker Swarm facilitate this process. For example, using `'docker-compose up --scale web=3'`

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

```
docker-compose up --scale web=3
```

## 27. What Is the Difference Between Docker Restart Policies "no", "on-failure," And "always"?

These restart policies will provide flexibility in managing the container behavior based on specific requirements. The restart policy **"no"** gives full control over restarts, **"on-failure"** handles irregular issues, and **"always"** will ensures the constant availability. Choose the appropriate policy based over the nature and importance of that particular containerized application.

Restart Policy	Description
<b>"no"</b>	<p>No automatic restart will be done. The container will not restart automatically, in any case of the exit status. It will be suitable for the scenarios where manual intervention is preferred or when the container is perfered for a one-time execution.</p> <p>Example: <b>docker run --restart no my_container</b></p>

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

Restart Policy	Description
"on-failure"	<p>It will restarts the container only if it exited with a non-zero status. This policy is useful when anticipating occasional failures and wanting the container to automatically recover from such failures.</p> <p>Example:</p> <pre><b>docker run --restart on-failure:3 my_container</b></pre> <p>( restarts up to 3 times on failure )</p>
"always"	<p>The container will restart regardless of its exit status. It will be useful in critical services that should be always running and ensuring continuous operation even if the container exits.</p> <p>Example:</p> <pre><b>docker run --restart always my_container</b></pre>

## 28. How Do You Inspect the Metadata of a Docker Image?

By using the ``docker inspect <image_name>`` command , you can examine into detailed metadata about the Docker image. This contains the information regarding labels, layers, and the configuration settings.

```
docker inspect <image_name>
```

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

Om using the **--cpus** option you can set the CPU limits and with **-m** option you can set memory limits. The following example illustrates usuage of CPU and memory for a docker container.

```
docker run --cpus=3 -m 1024M <image_name>
```

30. What are the Differences Between Docker Community Edition (CE) and Docker Enterprise Edition (EE)?

Usage of Docker Community Edition will be peferable for individuals and small-scale projects, It provides the essential features of containerization for free. On the other hand, Docker Enterprise Edition deals in providing the enterprise needs with advanced features and support for the large-scale projects in production environments. The choice between these two will depends on the scale, requirements, and support needed for the Docker deployment.

Feature	Docker Community Edition (CE)	Docker Enterprise Edition (EE)
Pricing	Free for individual use and will be suitable for development and testing.	It requires the subscription and offers advanced features for the production environments.
Support	Provides community support through forums and community resources	Provides enterprise-grade level support through service-level agreements (SLAs).

Feature	Docker Community Edition (CE)	Docker Enterprise Edition (EE)
		for a wide range of operating systems, cloud providers, and plugins.
Security	Contains Basic security features	Enhanced security with additional features like image signing and scanning are available.
Orchestration Tools	Has only Basic orchestration capabilities	Advanced orchestration tools like Docker Swarm and Kubernetes for large-scale deployments are available.
Image and Container Management	Core image and container management features.	Additional management tools and features, including role-based access control (RBAC).
Environment Support	Ideal for development and small-scale deployments.	Tailored for large-scale enterprise environments with optimized performance.
Networking		

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

Feature	Docker Community Edition (CE)	Docker Enterprise Edition (EE)
		multi-host networking and DNS.
<b>Plugins and Extensions</b>	Plugins and extensions are limited in the community edition.	A wide range support of certified plugins and extensions are available for various integrations.

## Advanced Docker Interview Questions for Experienced

### 31. What Is the Purpose of the "docker checkpoint" Command?

The **"docker checkpoint"** command is vital for the creation of snapshots of a running container's state, including its file system and the memory. It is particularly useful for experimental mode of scenarios such as debugging or migration.

For example to checkpoint a container named "my\_container," the command would be:

```
docker checkpoint create my_container checkpoint_name
```

### 32. Can We Use JSON Instead Of YAML While Developing a Docker-Compose File in Docker?

Yes, Docker Compose has support for both YAML and JSON formats for

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

preference on requirements of the projects. To use JSON, simply try on creating a ``docker-compose.json`` file instead of a ``docker-compose.yml`` file, and define your services in JSON format.

### 33. Describe the Lifecycle of a Docker Container.

In the lifecycle of docker container , it goes through the following states:

- **Creation:** On using ``docker run`` container is created.
- **Execution:** In this state specific required commands are executed inside the container.
- **Pausing/Unpausing:** These are optional states for temporarily halting a container.
- **Stopping:** On using the ``docker stop`` command container is gracefully halted in this state.
- **Removal:** Using ``docker rm`` command the container can be deleted.
- **Restarting:** With the ``docker restart`` command containers can be restarted.

### 34. How Will You Ensure Container 1 Runs Before Container 2 While Using docker-compose?

In Docker Compose, the order of the services startup is determined by their dependencies. By specifying container dependencies with the **"depends\_on"** key in the `docker-compose.yml` file, you can ensure the desired startup order.

A sample example on usage of `depends_on` provided here , In this even though `container1` is listed first because of the **`depends_on`** key `container2` will be startup and then `container1` will be queue order.

```
services:
  container1:
```

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).



```
container2:
```

```
....
```

### 35. How Does Docker Handle Container Isolation and Security?

Docker uses containerization concept to isolate the processes by limiting their access to the host system. Features like namespaces and cgroups provides the resource isolation for the containers and Docker Security Scanning helps in identifying the vulnerabilities in images.

### 36. How do the Docker Daemon and the Docker Client Communicate With Each Other?

The Docker daemon and client communicate on using REST APIs. The [Docker client](#) will send the commands to the daemon using the API, and the daemon will execute those commands on managing containers, images, and other Docker objects.

### 37. Can You Implement Continuous Development (CD) and Continuous Integration (CI) in Docker?

Yes, Docker is an integral part to the [CI/CD pipelines](#). Developers can use Docker images for the consistent environments, and CI tools can perform the automate testing and deployment using Docker containers for ensure reproducibility.

### 38. Is it a Good Practice to Run Stateful Applications on Docker?

Docker is primarily designed for the stateless applications. On using Docker volumes or persistent storage stateful applications can be runnable but it's crucial to carefully manage data persistence and backup to avoid

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

Docker secrets are used mostly to securely store sensitive information, such as passwords or API keys in Docker swarm. Secrets are encrypted and can only be accessible by services that have explicit permission to use them.

Example:

```
docker secret create db_password mysecretpassword
```

## 40. How Do You Create a Multi-stage Build in Docker?

A multi-stage build in Docker involves with using multiple "FROM" instructions in a Dockerfile. Each "FROM" instruction will begin a new stage, allowing you to build and copy the artifacts from previous stages for reducing the final image size.

Example of Dockerfile with multi-stage build:

```
FROM builder as build
# Build stage

FROM alpine
# Final stage
COPY --from=build /app /app
```

## 41. How Do You Update a Docker Container Without Losing Data?

To update a Docker container without losing data, you can try on using a combination of Docker volumes or bind mounts to make the data persistent outside the container. When updating, create a new container with the updated image and then link it to the existing data volume.

## 42. How Do You Manage Network Connectivity Between Docker Containers And the Host Machine?

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

Docker provides several ways to manage network connectivity between containers and the host machine. The choice of networking options depends on the specific requirements of the application and the desired level of isolation.

- **Bridge Networks:** Bridge networks are the default networks, these are created when a Docker daemon starts. Through this network containers on the same bridge network can communicate with each other.
- **Host Networks:** In this containers will share the host's network namespace. so that containers can directly use the host machine's network interfaces.
- **Custom Networks:** Custom bridge networks can be used to create isolated containers and control their communication. Containers on custom networks can communicate with each other with the help of container names as hostnames.
- **Overlay Networks (Swarm Mode):** Overlay networks are used in Docker Swarm mode for communication between the services that are running on different nodes. They provide multi-host networking for orchestration of containers.
- **Macvlan Networks:** Macvlan networks will allow containers to have their own MAC addresses on the physical network, providing them with the direct access to the host's network.

Examples for creating the networks are listed here:

```
# Create a bridge network
```

```
docker network create my_bridge_network
```

```
# To Run a container with host network
```

```
docker run --name container1 --network host -d my_image
```

```
# To create a custom bridge network
```

```
docker network create mv custom_network
```

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

```
# To create an overlay network
```

```
docker network create --driver overlay my_overlay_network
```

### 43. How Do You Debug Issues in a Docker Container?

Debugging techniques will provide a comprehensive approach for troubleshoot and to the resolve issues within Docker containers. Depending on the nature of the problem on following these guided commands you can understands the details of the container's behavior.

- **Container Logs:** On running this `'docker logs <container_id>'` command you can view the standard output and error logs.

```
docker logs <container_id>
```

- **Interactive Shell:** Helps in accessing the interactive shell inside the container for detailing.

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

- **Inspect Container Details:** helps in retrieving the detailed information about the container.

```
docker inspect <container_id>
```

- **Process Listing:** Provide the list of running processes inside the container.

```
docker exec -it <container_id> ps aux
```

- **Network Troubleshooting:** Check network connectivity of the container through the hostname

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

- **Resource Utilization:** Helps in monitoring the CPU and memory usage.

```
docker stats <container_id>
```

#### 44. How Does Docker Handle Service Discovery in Swarm Mode?

In Docker Swarm Mode, service discovery is automatically handled through maintaining an internal [DNS service](#) that automatically assigns DNS names to the containers on enabling easy service discovery within the swarm.

### Scenario-Based Docker Interview Questions and Answers

#### 45. Which Is the Preferred Method for Removing a Container: Using the "docker stop" Command Followed By the "docker rm" Command, or Simply Using the "docker rm" Command By Itself?

The recommend approach for the container removal is to use the combined "**docker stop**" and "**docker rm**" commands, as it makes sure a safely stopping of the container before removing it. This two-step process will helpful in avoiding potential issues related to the active processes within the container. However, if you are sure about that the container is not running, then you can go for using the "docker rm" command alone to remove it.

#### 46. Suppose You Have 3 Containers Running, and Out Of These, You Wish to Access One Of Them. How Do You Access a Running Container?

To access the running container you can use the "**docker exec**" command

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

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

The `docker exec` command is useful for the execute of a command inside a running container. It provides a interactive session for pseudo terminal - **TTY** with option `-it`, allowing you to interact with the container `<container_id_or_name>`. Try on replacing this with the actual ID or name of the container that you want to access.

The `/bin/bash` program specifies the command that to be executed in the container. In this case, it starts with interactive Bash shell, but you can also replace it with the appropriate required command for your needs. This will opens up the bash shell inside the specified container on enabling you to run commands, inspecting the container's filesystem, or troubleshoot. After once you are done with it, you can exit from the shell and it won't affect the container's running state.

#### 47. Considering a Server With 16 GB RAM and a Quad-core CPU, What Factors Determine the Maximum Number Of Containers You Can Run on the Host for a Microservices App?

The maximum number of containers a host can support will mainly depends on the available resources like RAM and CPU cores. With the 16 GB of RAM, on assuming each container utilizes 512 MB efficiently, you could potentially run on around nearly 32 containers. However, cautious allocation, monitoring tools, and the container orchestration are essential factors for optimal resource utilization and scalability.

#### 48. How Will You Monitor Docker in Production?

For monitoring the docker in production environment, generally utilize tools like Docker Stats, cAdvisor, and the [Prometheus](#) for real-time insights retrieving for container performance. Implementation of centralized

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

Additionally, consideration of using container orchestration platforms like [Kubernetes](#) or Docker Swarm, which offer better built-in monitoring and scaling capabilities. Regularly review metrics such as [CPU](#), memory usage, and network activity to ensure optimal performance and address potential issues promptly.

#### 49. Are You Aware of Load Balancing Across Containers And Hosts? How Does It Work?

[Load balancing](#) across the containers and hosts is critical for distributing traffic efficiently in the containerized environment. Container orchestration tools like Kubernetes or Docker Swarm employ load balancers to evenly distribute the requests among the container instances or nodes.

This will enhance the scalability, fault tolerance, and resource utilization by directing the traffic to healthy containers or hosts. The Load balancing plays a vital role in maintaining the stability and optimization on overall system performance in both the dynamic and scalable containerized applications.

#### 50. How Do You Share Data Between Containers in Docker?

In Docker, you can share data between the containers on using **volumes** or by utilizing the `--volumes-from` option. Volumes will provide a persistent and the shared storage mechanism, allowing the data to be accessed and modified by multiple containers.

Alternatively, the `--volumes-from` option allows the container to access the volumes of another container. This provides the seamless data sharing and collaboration between the containers, through facilitating the communication and coordination in complex multi-container setups.



Yes, The containers can be configured to restart by themselves automatically. Docker provides a restart policy that allowing you to define the container behavior when it exits. On using the options such as **"--restart always"**.

you can instruct Docker to restart the container automatically, in any case of the exit status. This will be useful for ensuring the continuous availability of critical services within the containerized environment. Other restart policies include **"unless-stopped"** and **"on-failure"** offering the flexibility in handling the container lifecycle events.

## 52. How Do You Perform a Live Migration Of Docker Containers Between Hosts?

For performing a live migration of docker containers between hosts can be achieved through using container orchestration tools like Docker Swarm or Kubernetes. These tools will manage the seamless containers movements across the hosts on ensuring minimal downtime.

Through Utilizing the features such as Swarm's **'docker service update'** or Kubernetes' **'kubectl drain'** and **'kubectl uncordon'** commands, you can initiate the live migrations by allowing containers to be moved to different hosts while maintaining the availability of application.

## Conclusion

In conclusion, preparing well for Docker interview questions is important for Java developers and all developer engineers, whether you're just starting out or have experience. This guide helps you understand Docker better, so you can show your skills in interviews and boost your career in tech, no matter your level.



## Top 50+ Docker Interview Questions and Answers (2024)

### Similar Reads

1. DevOps Interview Questions and Answers 2025
2. Terraform Interview Questions and Answers
3. Kubernetes Interview Questions and Answers
4. Virtualisation with Docker Containers
5. Top 10 Docker Container That Every Developer Should Know
6. How to Manage Docker Containers?
7. Best Practices for Docker Image Management and Versioning
8. Different Types of Docker Services
9. Top 20 Kubernetes Tools To Use in 2025
10. Docker Cheat Sheet : Complete Guide (2024)



#### Corporate & Communications Address:

A-143, 7th Floor, Sovereign Corporate Tower, Sector- 136, Noida, Uttar Pradesh (201305)

#### Registered Address:

K 061, Tower K, Gulshan Vivante Apartment, Sector 137, Noida, Gautam Buddh Nagar, Uttar Pradesh, 201305



We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

About Us  
Legal  
Privacy Policy  
Careers  
In Media  
Contact Us  
Corporate Solution  
Campus Training Program

Job-A-Thon  
Offline Classroom Program  
DSA in JAVA/C++  
Master System Design  
Master CP  
Videos

## Tutorials

Python  
Java  
C++  
PHP  
GoLang  
SQL  
R Language  
Android

## Data Science & ML

Data Science With Python  
Machine Learning  
ML Maths  
Data Visualisation  
Pandas  
NumPy  
NLP  
Deep Learning

## Python Tutorial

Python Examples  
Django Tutorial  
Python Projects  
Python Tkinter  
Web Scraping  
OpenCV Tutorial  
Python Interview Question

## DevOps

Git  
AWS

## DSA

Data Structures  
Algorithms  
DSA for Beginners  
Basic DSA Problems  
DSA Roadmap  
DSA Interview Questions  
Competitive Programming

## Web Technologies

HTML  
CSS  
JavaScript  
TypeScript  
ReactJS  
NextJS  
NodeJs  
Bootstrap  
Tailwind CSS

## Computer Science

GATE CS Notes  
Operating Systems  
Computer Network  
Database Management System  
Software Engineering  
Digital Logic Design  
Engineering Maths

## System Design

High Level Design  
Low Level Design

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

[DevOps Roadmap](#)[System Design Bootcamp](#)[Interview Questions](#)

## School Subjects

Mathematics  
Physics  
Chemistry  
Biology  
Social Science  
English Grammar

## Databases

SQL  
MYSQL  
PostgreSQL  
PL/SQL  
MongoDB

## Preparation Corner

Company-Wise Recruitment Process  
Aptitude Preparation  
Puzzles  
Company-Wise Preparation

## More Tutorials

Software Development  
Software Testing  
Product Management  
Project Management  
Linux  
Excel  
All Cheat Sheets

## Courses

IBM Certification Courses  
DSA and Placements  
Web Development  
Data Science  
Programming Languages  
DevOps & Cloud

## Programming Languages

C Programming with Data Structures  
C++ Programming Course  
Java Programming Course  
Python Full Course

## Clouds/Devops

DevOps Engineering  
AWS Solutions Architect Certification  
Salesforce Certified Administrator Course

## GATE 2026

GATE CS Rank Booster  
GATE DA Rank Booster  
GATE CS & IT Course - 2026  
GATE DA Course 2026  
GATE Rank Predictor

---

@GeeksforGeeks, Sanchhaya Education Private Limited, All rights reserved

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).