

# Docker Interview Question

## 1. What is the Difference between an Image, Container, and Engine?

### Image:

An image is a lightweight, standalone, executable software package containing everything needed to run a piece of software. 🖼️

### Container:

A container is an instance of an image, encapsulating application code, runtime, libraries, and system tools, ensuring consistent behaviour across different environments. 📦

### Engine:

The Docker Engine is the core component responsible for building, running, and managing containers. It includes a server, REST API, and command-line interface. 🛠️

## 2. What is the Difference between the Docker command COPY vs ADD?

**COPY:** The `COPY` command in a Dockerfile is used to copy files from the host system to the container. 📄

**ADD:** The `ADD` command not only copies files but can also fetch files from URLs and extract compressed files. It's versatile but uses it judiciously to avoid unexpected behaviour. 🚰

## 3. What is the Difference between the Docker command CMD vs RUN?

**CMD:** The `CMD` instruction defines the default command and arguments for a container, which can be overridden when starting the container. 🚀

**RUN:** The `RUN` instruction executes commands during the image build process to set up the environment, install packages, and configure the container. 🧑






## 4. How Will you reduce the size of the Docker image?

### Optimise your Docker images by:



- Using multi-stage builds to include only necessary artefacts. 🏠
- Choosing lightweight base images. ☐
- Minimising layers by chaining commands and cleaning up. 🧹
- Removing unnecessary files and caches. 🗑️

## 5. Why and when to use Docker?



### Use Docker for:



- Consistency: Ensure that applications run the same way everywhere. 
- Isolation: Avoid conflicts between applications by using containers.  
- Efficiency: Share and deploy applications faster with containers. 
- Scalability: Seamlessly scale applications up or down. 


## 6. Explain the Docker components and how they interact with each other.



Docker components include the Docker Client, API, Daemon, Images, Containers, and Registry. The Client communicates with the Daemon, which builds, runs, and manages Containers using Images. The Registry stores and distributes Images.  

## 7. Explain the terminology: Docker Compose, Docker File, Docker Image, Docker Container?



Docker Compose: A tool to define and manage multi-container applications using a YAML file.  

Docker File: A script containing instructions to build a Docker Image.  




Docker Image: A snapshot of a containerized application along with its environment and dependencies. 

- **\*\*Docker Container:\*\*** An isolated runtime instance of an Image, running as a process on the host.  

## 8. In what real scenarios have you used Docker?







Share your experiences of using Docker to streamline development, create consistent testing environments, and simplify deployment processes.  

## 9. Docker vs Hypervisor?




Docker containers share the host OS kernel, making them lightweight and efficient, while hypervisors provide full isolation by emulating hardware. Containers offer faster startup times and better resource utilisation compared to hypervisors.   

## 10. What are the advantages and disadvantages of using Docker?



### Advantages:

- Consistency across environments.  
- Efficient resource utilisation.  
- Rapid application deployment. 
- Isolation and security. 



### Disadvantages:

- Limited for certain use cases. 
- Complexity in managing orchestration. 
- Security concerns if not properly configured. 



## 11. What is a Docker namespace?

A namespace in Docker provides process isolation, allowing each container to have its own view of the system resources, like PIDs, network, and user IDs.  

## 12. What is a Docker registry?







A Docker registry is a repository that stores Docker Images. It allows you to share, distribute, and manage Images, both publicly and privately.  

## 13. What is an entry point?

The `ENTRYPOINT` in a Dockerfile specifies the default command that will be executed when a container starts. It's like the main command that defines the container's primary purpose.  

## 14. How to implement CI/CD in Docker?

Integrate Docker into your CI/CD pipeline by:

- Creating Dockerfiles for building Images.  
- Automating Image builds and pushes to a Docker registry.  
- Orchestrating deployment and updates using tools like Kubernetes.  

**15. Will data on the container be lost when the docker container exits?**

Yes, unless data is persisted using volumes or bind mounts, data within a container will be lost when it exits. ⚠️📁

**16. What is a Docker swarm?**

A Docker swarm is a clustering and orchestration solution that allows you to manage multiple Docker nodes as a single entity, enabling high availability and scaling. 🐄🐑

**17. What are the docker commands for the following:**

- View running containers:  
`docker ps` 📁👤
- Command to run the container under a specific name:  
`docker run --name <container\_name> <image\_name>` 🧑🔥
- Command to export a Docker:  
`docker export <container\_id> > <output\_file>.tar` 📁📦
- Command to import an already existing Docker image:  
`docker import <input\_file>.tar` 📁📦
- Commands to delete a container:  
`docker rm <container\_id>` 🗑️
- Command to remove all stopped containers, unused networks, build caches, and dangling images:  
`docker system prune` 🗑️🧹

**18. What are the common Docker practices to reduce the size of Docker Image?**

- Use a minimal base image. 📁🔍
- Combine commands in a single `RUN` instruction. 🎯🧑
- Remove unnecessary files and dependencies. 🚫📁
- Leverage multi-stage builds. 🧑🏠
- Minimise the number of layers. 📁📊

Mastering Docker basics opens doors to efficient software development and deployment. Remember, with great power comes great