# Important Docker Interview Questions

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

- Docker Image: A lightweight, standalone, and immutable file that contains the application and its dependencies.
  - Docker Container: A runtime instance of a Docker Image. It is an isolated environment where the application runs.
  - Docker Engine: The core component of Docker, responsible for building, running, and managing containers.

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

- COPY: Copies files and directories to the Docker image. It is simple and predictable.
  - ADD: Extends COPY by allowing:
    - Extraction of TAR files.
    - Downloading files from remote URLs.
  - Best Practice: Use `COPY` unless you specifically need `ADD` features.

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

- CMD: Specifies the default command to run when the container starts. It is overridden when a command is passed during `docker run`.
  - RUN: Executes a command during the image build process and commits the result in the image.

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

- Use multi-stage builds.
  - Combine `RUN` commands to reduce intermediate layers.
  - Clean up temporary files in each layer.
  - Use a minimal base image, e.g., `alpine`.

## 5. Why and when should you use Docker?

- Simplifies application deployment by packaging applications with their dependencies.
  - Ensures consistency across environments.
  - Reduces overhead compared to traditional virtual machines.
  - Ideal for CI/CD pipelines, microservices architecture, and testing environments.

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

- Docker Client: CLI to interact with the Docker daemon.
  - Docker Daemon: Background service managing images, containers, and networks.

- Docker Registry: Stores Docker images (e.g., Docker Hub).
- Docker Objects: Include Images, Containers, Volumes, and Networks.

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

- Docker Compose: A tool to define and run multi-container applications.
  - Dockerfile: A script containing instructions to build a Docker image.
  - Docker Image: A lightweight file that contains the application and dependencies.
  - Docker Container: A runtime instance of a Docker image.

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

- Automating CI/CD pipelines.
  - Running microservices.
  - Isolating development environments.
  - Testing applications in isolated containers.

## 9. Docker vs Hypervisor?

- Docker: Lightweight, uses OS-level virtualization.
  - Hypervisor: Uses hardware virtualization; heavier than Docker.

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

- Advantages:
  - Portability.
  - Consistency.
  - Efficiency.
- Disadvantages:
  - Limited GUI applications.
  - Security concerns due to shared kernel.

## 11. What is a Docker namespace?

- A Linux kernel feature providing isolation for containers (e.g., process, network, mount, IPC namespaces).

## 12. What is a Docker registry?

- A centralized storage and distribution system for Docker images (e.g., Docker Hub, Azure Container Registry).

## 13. What is an entry point?

- The default application or process that runs when a container starts.

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

- Use Docker in CI/CD pipelines to:
  - Build images for each commit.
  - Test images in isolated environments.
  - Deploy containers as part of the pipeline.

### 15. Will data on the container be lost when the Docker container exits?

- Yes, unless data is stored in a volume or bind mount.

### 16. What is a Docker swarm?

- A native clustering and orchestration tool for Docker containers.

### 17. What are the Docker commands for the following:

- Viewing running containers: `docker ps`
- Running a container under a specific name: `docker run --name container_name image`
- Exporting a Docker image: `docker save -o image.tar image_name`
- Importing an existing Docker image: `docker load -i image.tar`
- Deleting a container: `docker rm container_name`
- Removing 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 images?

- Use a minimal base image.
  - Combine commands into a single `RUN` instruction.
  - Use `.dockerignore` to exclude unnecessary files.

### 19. How do you troubleshoot a Docker container that is not starting?

- Use `docker logs container_name` to view logs.
  - Inspect the container with `docker inspect`.
  - Check container events using `docker events`.

### 20. Can you explain the Docker networking model?

- Docker provides three main networks:
  - Bridge: Default network for containers.
  - Host: Shares the host's network stack.
  - None: No networking.

### 21. How do you manage persistent storage in Docker?

- Use volumes: `docker volume create`.
  - Use bind mounts for specific host paths.

### 22. How do you secure a Docker container?

- Use minimal base images.
  - Limit container privileges.
  - Use signed images.
  - Scan images for vulnerabilities.

### 23. What is Docker overlay networking?

- A Docker feature allowing communication between containers across multiple hosts in a Swarm.

## 24. How do you handle environment variables in Docker?

- Use `ENV` in a Dockerfile.
  - Use `-e` flag with `docker run`.
  - Use `.env` files with `docker-compose`.