**Docker Interview Questions**

**Basic Questions**

1. **What is Docker?**
2. Docker is an open-source platform that uses containerization to automate the deployment, scaling, and management of applications. It packages an application with its dependencies into a lightweight, portable container that runs consistently across different environments.
3. **What is the difference between a Docker container and a virtual machine (VM)?**
   * **Answer**: A VM runs a full guest OS with virtualized hardware, making it resource-heavy and slower to start. A Docker container shares the host OS kernel and only includes the application and its dependencies, making it lightweight and faster.
4. **What is a Docker image?**

A Docker image is a read-only template that contains the application, libraries, dependencies, and instructions needed to create a container.

1. **What is Docker Hub?**

Docker Hub is a cloud-based registry service for storing, sharing, and managing Docker images. It offers public repositories for community use and private repositories for secure storage.

1. **How do you list all running Docker containers?**

* docker ps # Lists running containers
* docker ps -a # Lists all containers (running & stopped)

**Intermediate-Level Questions**

1. **What is a Dockerfile and what is it used for?**

A Dockerfile is a script with instructions to build a Docker image. It includes commands like FROM (base image), RUN (execute commands), COPY (add files), and CMD (default container command). It automates image creation.

1. **What is the difference between CMD and ENTRYPOINT in a Dockerfile?**

CMD provides a default command or arguments for a container, which can be overridden when running the container.

ENTRYPOINT specifies a fixed executable that always runs, with arguments from CMD or the command line appended to it. Together, they define container behaviour. It makes the command non-overridable by default.

1. **How do you remove a Docker container?**
   * **Answer**: To remove a stopped container, use docker rm <container\_id>. If the container is running, stop it first with docker stop <container\_id> or force removal with docker rm -f <container\_id>.
2. **What is Docker Compose and when would you use it?**

Docker Compose is a tool for defining and running multi-container applications using a YAML configuration file (docker-compose.yml). It’s used to manage services, networks, and volumes, making it ideal for development, testing, and small-scale deployments.

1. **How do you persist data in Docker containers?**

Data persistence is achieved using Docker volumes or bind mounts. Volumes (e.g., docker volume create) are managed by Docker and stored in a host directory. Bind mounts (e.g., -v /host/path:/container/path) map a specific host directory to the container.

**Advanced-Level Questions**

1. **What is Docker Swarm?**

**Answer**: Docker Swarm is Docker’s native orchestration tool for managing a cluster of Docker nodes. It provides features like service scaling, load balancing, and high availability for deploying containerized applications across multiple hosts.

1. **How does Docker networking work?**

Docker supports several network drivers: bridge (default, for single-host container communication), host (uses the host’s network stack), and overlay (enables multi-host networking). Containers in the same user-defined network can communicate using their names or IPs.

1. **How can you optimize the size of a Docker image?**

To optimize image size, use a smaller base image (e.g., Alpine), combine RUN commands to reduce layers, clean up unnecessary files (e.g., rm -rf in the same layer), and use multi-stage builds to exclude build-time dependencies from the final image.

1. **What is a multi-stage build in Docker, and why is it useful?**
   * **Answer**: A multi-stage build uses multiple FROM statements in a Dockerfile to separate build and runtime environments. For example, one stage compiles the app, and another copies only the compiled output into a lean runtime image. It reduces image size by excluding build tools and intermediate files.

15**.Explain the main components of Docker.**

* + **Docker Engine**: The core component that runs and manages containers.
  + **Docker Image**: A lightweight, standalone package containing application code and dependencies.
  + **Docker Container**: A running instance of a Docker image.
  + **Docker Registry**: A repository to store and share images (e.g., Docker Hub).
  + **Docker Compose**: A tool to manage multi-container applications.

16.**What is a Docker image, and how is it different from a Docker container?**  
A Docker image is a read-only template used to create containers. A Docker container is a running instance of an image.

17**.How do you create and run a Docker container?**

docker run -d --name my\_container my\_image

18.**What are the different ways to create a Docker image?**

* + Using a **Dockerfile** and docker build
  + Committing changes from a running container with docker commit
  + Pulling from a Docker registry like Docker Hub

19**.What is a Dockerfile, and why is it used?**  
A Dockerfile is a script containing instructions to build a Docker image automatically.

20**.How do you build a Docker image from a Dockerfile?**

* docker build -t my\_image.

21.**How do you restart a Docker container?**

docker restart my\_container

22**.How can you copy files into a Docker container?**

docker cp myfile.txt my\_container:/path/

23**.What is Docker Compose, and when would you use it?**  
 Docker Compose is used to manage multi-container applications with a YAML configuration file.

24**.Explain Docker networking and its types.**

* + **Bridge**: Default network for standalone containers.
  + **Host**: Shares the host network stack.
  + **Overlay**: Used in Docker Swarm for multi-host communication.
  + **None**: No network.

25**.What are Docker volumes, and why are they important?**  
Volumes store data persistently outside of a container’s filesystem.

26.**How do you optimize Docker images for production?**

* + Use multi-stage builds.
  + Minimize layers and use smaller base images.
  + Clean up unnecessary files.

27**.How does Docker handle security, and what best practices should be followed?**

* + Use minimal base images.
  + Scan images for vulnerabilities.
  + Implement the principle of least privilege.

28.**What is Kubernetes, and how does it relate to Docker?**  
Kubernetes is an orchestration platform for managing containerized applications at scale.

29**.Explain how to handle environment variables in Docker.**

docker run -e MY\_ENV\_VAR=value my\_image