DevOps Shack

Top 50 Docker Interview Questions and Answers asked in MNC interviews

1. What is Docker?

Answer:
Docker is an open-source containerization platform that enables developers to package applications along with their dependencies into lightweight, portable containers. These containers can run seamlessly across different computing environments.

2. What is a Docker container?

Answer:
A Docker container is a lightweight, standalone, and executable package that includes everything needed to run an application—code, runtime, libraries, and settings.

3. How do you reduce the size of a Docker image?

Answer:

- **Use minimal base images, like `alpine` or `distroless`.**
- **Minimize the number of layers in the `Dockerfile`.**
- **Use `.dockerignore` to exclude unnecessary files.**

Combine `RUN` statements:

```
RUN apt-get update && apt-get install -y \
```

- `curl && rm -rf /var/lib/apt/lists/*`

4. Explain Docker architecture.

Answer:
Docker architecture consists of:

- **Docker Client: The CLI or GUI tool to interact with Docker.**
- **Docker Daemon: Runs on the host and manages containers, images, and volumes.**
- **Docker Images: Read-only templates to create containers.**
- **Docker Registries: Store and distribute Docker images.**
- **Containers: Runtime instances of images.**

5. What is a Docker image?

Answer:
A Docker image is a read-only template that contains the application, runtime environment, and dependencies required to run it. It serves as a blueprint for containers.

6. How do you create a Docker image?

Answer:
You can create a Docker image using:

A Dockerfile (recommended):

```
FROM node:16
WORKDIR /app
COPY . .
RUN npm install
```

```
CMD ["node", "app.js"]
```

- **Using the docker commit command (not recommended for production).**

7. What is a Dockerfile?

Answer:
A Dockerfile is a script-like text document containing a sequence of commands to build a Docker image. Example:

```
FROM python:3.8

WORKDIR /app
COPY requirements.txt .
RUN pip install -r requirements.txt
COPY . .
CMD ["python", "app.py"]
```

8. What is the difference between `docker-compose up` and `docker-compose run`?

Answer:

- `docker-compose up`**: Builds, starts, and runs all the services defined in** `docker-compose.yml`**.**
- `docker-compose run`**: Runs a single, one-off service without starting the entire application stack.**

9. What is Docker Compose?

Answer:

Docker Compose is a tool for defining and managing multi-container applications using a YAML file (`docker-compose.yml`).

Example:

```
version: '3'

services:
  web:
    image: nginx
    ports:
      - "80:80"
```

10. How do you expose a container's port?

Answer:
Use the `-p` or `--publish` flag:

```
docker run -d -p 8080:80 nginx
```

11. What is BuildKit in Docker?

Answer:
BuildKit is a modern backend for building Docker images that improves build performance and efficiency. It supports parallel builds, caching, and output control.

Enable BuildKit:

```
DOCKER_BUILDKIT=1 docker build .
```

12. What is a Docker volume?

Answer:

A volume is a storage mechanism to persist data generated by containers. Volumes exist outside the container's lifecycle.

13. How do you create and manage Docker volumes?

Answer:

Create:

```
docker volume create my-volume
```

Attach to a container:

```
docker run -v my-volume:/data nginx
```

14. What is Docker Swarm?

Answer:

Docker Swarm is Docker's native orchestration tool that allows clustering of multiple Docker hosts.

15. How do you scale containers in Docker Compose?

Answer:
Use the `--scale` flag:

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

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

Answer:
Multi-stage builds reduce image size by using multiple `FROM` statements. Example:

```
# Build Stage
FROM node:16 as builder
WORKDIR /app
COPY package.json .
RUN npm install

# Final Stage
FROM nginx
COPY --from=builder /app /usr/share/nginx/html
```

17. How do you troubleshoot a failing container?

Answer:

- Check logs: `docker logs <container_id>`
- Access container: `docker exec -it <container_id> bash`
- Inspect: `docker inspect <container_id>`

18. What is the purpose of `.dockerignore`?

Answer:
The `.dockerignore` file excludes specific files or directories from being copied to the Docker image.

19. What is Docker Hub?

Answer:
Docker Hub is a cloud-based registry to store and distribute Docker images.

20. How do you push an image to Docker Hub?

Answer:

```
docker tag my-app:latest username/my-app:latest
docker push username/my-app
```

21. What are namespaces in Docker?

Answer:
Namespaces isolate processes, networking, and file systems for containers.

22. What are cgroups in Docker?

Answer:
Control Groups (cgroups) limit and monitor resource usage like CPU, memory, and disk I/O for containers.

23. How do you secure a Docker container?

Answer:

- **Use minimal base images (e.g., `alpine`).**
- **Run containers with non-root users.**
- **Scan images for vulnerabilities using tools like Trivy.**

24. What is the purpose of `docker network`?

Answer:
`docker network` **manages communication between containers.**

25. How do you list all running containers?

Answer:

```
docker ps
```

26. How do you stop and remove containers?

Answer:

```
docker stop <container_id>
docker rm <container_id>
```

27. What are the types of Docker networks?

Answer:

- **Bridge: Default for standalone containers.**
- **Host: Shares the host's network.**
- **None: No networking.**
- **Overlay: For multi-host setups.**

28. How do you build a Docker image?

Answer:

```
docker build -t my-app .
```

29. How do you inspect a Docker image?

Answer:

```
docker inspect <image_id>
```

30. How do you clean up unused Docker resources?

Answer:

```
docker system prune
```

31. What is the difference between `docker run` and `docker start`?

Answer:

- `docker run`: Creates and starts a new container.
- `docker start`: Restarts an existing container.

32. What is `docker exec`?

Answer:
Runs a command in a running container:

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

33. How do you check the resource usage of a container?

Answer:

```
docker stats
```

34. What is a dangling image?

Answer:
An image without a tag, often left after rebuilding.

35. How do you remove a Docker image?

Answer:

```
docker rmi <image_id>
```

36. What is the default storage driver in Docker?

Answer:
On Linux, it is `overlay2`.

37. What is the purpose of `docker tag`?

Answer:
Tags an image with a name and version:

```
docker tag my-app:latest my-repo/my-app:v1
```

38. What is the difference between `bind mounts` and `volumes`?

Answer:

- **Bind mounts: Tied to the host filesystem.**
- **Volumes: Managed by Docker and portable.**

39. How do you list all Docker networks?

Answer:

```
docker network ls
```

40. What is Docker's default restart policy?

Answer:
`no`**: Containers do not restart unless manually done.**

41. How do you pass environment variables to a container?

Answer:

```
docker run -e VAR_NAME=value my-app
```

42. How do you enable auto-restart for containers?

Answer:

```
docker run --restart=always my-app
```

43. What are overlay networks in Docker?

Answer:

Used for communication between Swarm services.

44. What is the `docker inspect` command used for?

Answer:
To view detailed information about containers or images.

45. How do you check Docker logs?

Answer:

```
docker logs <container_id>
```

46. What is a health check in Docker?

Answer:
Used to determine if a container is healthy:

```
HEALTHCHECK CMD curl -f http://localhost || exit 1
```

47. What is Docker Registry?

Answer:
A service to store Docker images (e.g., Docker Hub, ECR).

48. How do you debug a Docker build?

Answer:

- Use `docker build --progress=plain`.
- Inspect intermediate layers with `docker history`.

49. What is the difference between `RUN`, `CMD`, and `ENTRYPOINT` in Dockerfile?

Answer:

- **RUN: Executes commands during image build.**
- **CMD: Default command executed during container runtime.**
- **ENTRYPOINT: Executable during runtime.**

50. What tools integrate with Docker?

Answer:

- **CI/CD Tools: Jenkins, GitLab CI/CD.**
- **Monitoring: Prometheus, Grafana.**
- **Orchestration: Kubernetes, Docker Swarm.**