DevOps Shack

100 Docker Questions and Answers Asked in MNC Interviews

1. What is Docker, and how does it work?

Answer: Docker is a platform that allows developers to build, ip, and run applications in containers. Containers are lightweight, portable, and consistent across different environments. Docker packages an application with all its dependencies, libraries, and configuration files into a container image. Containers use the host OS kernel, making them more lightweight compared to virtual machines.

2. What are containers, and how do they differ from virtual machines?

Answer: Containers are lightweight, standalone executable packages that include everything needed to run a piece of software. Unlike virtual machines, which include a full OS, containers are the host OS kernel, making them more efficient in terms of resource utilization.

3. What are Docker images, and how do they differ from containers?

Answer: Docker images are read-only templates that contain the application code, runtime, libraries, and dependencies needed to run an application. Containers are instances of these images that are run on the Docker platform.

4. What is the difference between Docker CE and Docker EE?

Answer: Docker CE (Community Edition) is a free, open-source version of Docker

suitable for individuals and small teams. Docker EE (Enterprise Edition) is a paid version designed for large organizations and includes additional security, support, and enterprise-grade features.

5. What is Docker Hub?

Answer: Docker Hub is a public registry provided by Docker where users can are, store, and distribute container images. It serves as a central repository for images created by Docker users and official images maintained by Docker.

6. Explain the Dockerfile and its purpose.

Answer: A Dockerfile is a script containing instructions on how to build a Docker image. It includes commands to install dependencies, copy files, and define how the application ould run within a container. It allows developers to automate the image creation process.

7. What are the key instructions in a Dockerfile?

Answer: Common Dockerfile instructions include:

- `FROM`: Specifies the base image.
- `RUN`: Executes commands during the image build process.
- `COPY`/`ADD`: Copies files into the image.
- `CMD`: Defines the default command to run when the container starts.
- `ENTRYPOINT`: Defines the main executable for the container.
- `EXPOSE`: Exposes a port for network connections.
- `WORKDIR`: Sets the working directory inside the container.

8. What is the difference between CMD and ENTRYPOINT?

Answer: Both `CMD` and `ENTRYPOINT` define commands to be executed when a container starts. The difference is:

- **`CMD` provides default arguments for an entrypoint.**
- **`ENTRYPOINT` specifies a command that always runs, and any arguments passed to the container will be appended.**

9. How do you list running Docker containers?

Answer: Use the command:

```
docker ps
```

This ows a list of all running containers, including container IDs, names, and status.

10. How do you stop and remove a running container?

Answer: To stop a container, use:

```
docker stop <container_name_or_id>
```

To remove the container, use:

```
docker rm <container_name_or_id>
```

11. How do you remove all stopped containers?

Answer: Use the command:

```
docker container prune
```

This removes all stopped containers.

12. How do you remove an image from the local system?

Answer: Use the command:

```
docker rmi <image_name_or_id>
```

To force removal, use:

```
docker rmi -f <image_name_or_id>
```

13. What is the purpose of Docker Compose?

Answer: Docker Compose is a tool used to define and manage multi-container Docker applications. It uses a `docker-compose.yml` file to define services, networks, and volumes, making it easy to start, stop, and manage containers as a group.

14. How do you build and run a Docker Compose application?

Answer:

Build the application using:

```
docker-compose build
```

Start the services using:

```
docker-compose up
```

15. What is the difference between `docker-compose up` and

`docker-compose up -d`?

Answer:

- **`docker-compose up`: Starts services in the foreground.**
- **`docker-compose up -d`: Starts services in detached mode (in the background).**

16. How do you scale services in Docker Compose?

Answer: Use the command:

```
docker-compose up --scale <service_name>=<number>
```

Example:

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

17. What are Docker volumes, and why are they used?

Answer: Docker volumes are used to persist data generated and used by Docker containers. They provide a way to are data between containers and the host system, ensuring data persistence even when containers are deleted or restarted.

18. What is the difference between bind mounts and Docker volumes?

Answer:

- **Bind mounts: Link a specific host directory to a container directory.**
- **Volumes: Managed by Docker and stored in a central location on the host system.**

19. How do you create a volume and attach it to a container?

Answer: Create a volume using:

```
docker volume create my_volume
```

Attach it using:

```
docker run -v my_volume:/data <image_name>
```

20. What is the difference between the `COPY` and `ADD` instructions in a Dockerfile?

Answer:

- **`COPY`: Copies files/directories from the host to the image.**
- **`ADD`: Supports additional features like extracting archives from a URL, as well as copying.**

21. How do you restart a stopped Docker container?

Answer:
To restart a stopped container, use:

```
docker start <container_name_or_id>
```

To restart a running container:

```
docker restart <container_name_or_id>
```

22. What is the purpose of the `docker exec` command?

Answer:

The `docker exec` command is used to run additional commands in a running container. For example:

```
docker exec -it <container_name_or_id> /bin/
```

This opens an interactive ell inside the container.

23. What are the different networking modes in Docker?

Answer:

1. **Bridge (default for standalone containers): Containers connect through an isolated network bridge.**
2. **Host: Uses the host's networking stack.**
3. **None: No network interfaces; isolated from the network.**
4. **Overlay: Used for services in Docker Swarm.**
5. **Macvlan: Assigns a unique MAC address to the container.**

24. How do you create a user-defined bridge network?

Answer:

Use the command:

```
docker network create my_bridge
```

To attach a container:

```
docker run --network my_bridge <image_name>
```

25. How do you inspect a Docker container?

Answer:
To inspect container details, use:

```
docker inspect <container_name_or_id>
```

This provides detailed information about the container's configuration, networking, and state.

26. How do you limit a container's CPU and memory usage?

Answer:
To limit CPU usage:

```
docker run --cpus="1.5" <image_name>
```

To limit memory usage:

```
docker run -m="512m" <image_name>
```

27. What is Docker Swarm?

Answer:
Docker Swarm is Docker's native orchestration tool that allows you to manage a cluster of Docker nodes as a single virtual system, deploying services across multiple nodes for scalability and high availability.

28. How do you initialize a Docker Swarm?

Answer:
Use the command:

```
docker swarm init
```

To add nodes to the swarm, use the generated token displayed after initialization.

29. How do you deploy a service in Docker Swarm?

Answer:
To deploy a service, use:

```
docker service create --name <service_name> --replicas 3 <image_name>
```

30. What is the difference between a container and a service in Docker Swarm?

Answer:

- **Container: A standalone running instance of a Docker image.**
- **Service: Defines how containers run across a swarm cluster, managing scaling and load balancing.**

31. What are the states of a Docker container?

Answer:

1. **Created: Container is created but not started.**
2. **Running: Container is actively running.**
3. **Paused: Container execution is temporarily suspended.**
4. **Stopped/Exited: Container has stopped running.**

32. What is a multi-stage build in Docker, and why is it used?

Answer:

Multi-stage builds allow you to use multiple `FROM` statements in a Dockerfile to optimize image size. The final image only includes what's necessary to run the application, excluding build dependencies.

33. How do you use multi-stage builds in a Dockerfile?

Answer:
Example:

```
FROM golang:1.19 AS builder
WORKDIR /app
COPY . .
RUN go build -o main .

FROM alpine:latest
WORKDIR /app
COPY --from=builder /app/main .
CMD ["./main"]
```

34. How do you update a running container's configuration?

Answer:
You cannot directly update a running container's configuration. You must update the image or recreate the container with the desired configuration changes.

35. How do you check the logs of a container?

Answer:
Use the command:

```
docker logs <container_name_or_id>
```

To follow real-time logs:

```
docker logs -f <container_name_or_id>
```

36. How do you remove unused Docker images?

Answer:
Use the command:

```
docker image prune
```

To remove all unused images, dangling or not:

```
docker image prune -a
```

37. How do you create a Docker network?

Answer:
Use the command:

```
docker network create my_network
```

38. What is the purpose of the `EXPOSE` instruction in a Dockerfile?

Answer:
The `EXPOSE` instruction informs Docker that the container will listen on the specified port(s) at runtime. However, it does not publi the ports to the host system.

39. How do you publi ports when running a container?

Answer:

Use the `-p` option:

```
docker run -p 8080:80 <image_name>
```

This maps port 80 in the container to port 8080 on the host.

40. How do you copy files from a container to the host?

Answer:
Use the command:

```
docker cp <container_id>:<container_path> <host_path>
```

41. What is the difference between `docker stop` and `docker kill`?

Answer:

- `docker stop`: Gracefully stops a container, allowing it to terminate ongoing processes.
- `docker kill`: Immediately terminates a container without cleanup.

42. How do you check Docker's version?

Answer:
Use the command:

```
docker version
```

43. How do you check the storage usage of Docker?

Answer:
Use the command:

```
docker system df
```

44. How do you clean up all unused containers, networks, and images?

Answer:
Use the command:

```
docker system prune
```

45. What are Docker tags, and how do you create them?

Answer:
Docker tags are labels applied to images to identify different versions. To tag an image:

```
docker tag <image_id> <repository>:<tag>
```

46. How do you pu an image to Docker Hub?

Answer:

Login to Docker Hub:

```
docker login
```

Pu the image:

```
docker pu <repository>:<tag>
```

47. What is the default storage driver in Docker?

Answer:

On Linux, Docker commonly uses the overlay2 storage driver.

48. What is a dangling image in Docker?

Answer:
A dangling image is an image that has no tags and is not associated with any containers.

49. How do you run a container in interactive mode?

Answer:
Use the `-it` option:

```
docker run -it <image_name> /bin/
```

50. How do you forcefully remove a running container?

Answer:
Use the command:

```
docker rm -f <container_name_or_id>
```

51. How do you build a Docker image from a Dockerfile?

Answer:
Use the command:

```
docker build -t <image_name>:<tag> <path_to_dockerfile>
```

Example:

```
docker build -t myapp:latest .
```

52. How do you list all Docker volumes?

Answer:
Use the command:

```
docker volume ls
```

53. How do you inspect a Docker volume?

Answer:
Use the command:

```
docker volume inspect <volume_name>
```

54. How do you attach a container to an existing network?

Answer:
Use the command:

```
docker network connect <network_name>
<container_name_or_id>
```

55. How do you disconnect a container from a network?

Answer:
Use the command:

```
docker network disconnect <network_name>
<container_name_or_id>
```

56. How do you export and import a Docker container?

Answer:

To export a container:

```
docker export <container_name_or_id> > my_container.tar
```

To import it:

```
cat my_container.tar | docker import - my_image:latest
```

57. How do you rename a Docker container?

Answer:
Use the command:

```
docker rename <old_name> <new_name>
```

58. What is the difference between `docker run` and `docker create`?

Answer:

- `docker run`: Creates and starts a new container.
- `docker create`: Creates a container without starting it.

59. What are Docker secrets, and how are they used?

Answer:
Docker secrets are used to securely store sensitive information, such as passwords and API keys, in Docker Swarm. They are encrypted and only accessible by services that need them.

60. How do you create and use a Docker secret?

Answer:

To create a secret:

```
echo "mysecret" | docker secret create my_secret -
```

To use it in a service:

```
docker service create --secret my_secret <image_name>
```

61. What is Docker's overlay network?

Answer:
Overlay networks allow containers running on different Docker hosts to communicate with each other, enabling multi-host container communication in Swarm mode.

62. How do you set environment variables in a Docker container?

Answer:
Use the `-e` option:

```
docker run -e MY_VAR=value <image_name>
```

Or use the `ENV` instruction in a Dockerfile:

dockerfile

```
ENV MY_VAR value
```

63. What is the purpose of the `.dockerignore` file?

Answer:
The `.dockerignore` file specifies files and directories to exclude from the

Docker image build context, reducing image size and build time.

64. How do you view the history of a Docker image?

Answer:
Use the command:

```
docker history <image_name_or_id>
```

65. How do you check the status of a Docker service in Swarm mode?

Answer:
Use the command:

```
docker service ls
```

For detailed information:

```
docker service ps <service_name>
```

66. How do you restart all stopped containers?

Answer:
Use the command:

```
docker start $(docker ps -q -a)
```

67. What is the difference between `docker-compose down` and `docker-compose stop`?

Answer:

- `docker-compose stop`: **Stops the containers but preserves the**

network and volumes.

- `docker-compose down`: Stops and removes containers, networks, and optionally volumes.

68. How do you view the configuration of a running Docker service?

Answer:
Use the command:

```
docker service inspect <service_name>
```

69. How do you remove a Docker network?

Answer:
Use the command:

```
docker network rm <network_name>
```

70. How do you use Docker with Kubernetes?

Answer:
Docker is used as a container runtime in Kubernetes to manage and run containerized applications. Kubernetes pulls Docker images from registries to deploy them as pods.

71. How do you configure a Docker container to restart automatically?

Answer:
Use the `--restart` flag:

```
docker run --restart=always <image_name>
```

72. What is Docker Machine?

Answer:
Docker Machine is a tool used to provision Docker hosts on local or cloud environments. It allows you to create and manage multiple Docker hosts.

73. How do you display system-wide information about Docker?

Answer:
Use the command:

```
docker info
```

74. How do you change the default storage driver in Docker?

Answer:
Edit the Docker configuration file (`/etc/docker/daemon.json`) and set the `storage-driver`:

json

```
{
  "storage-driver": "overlay2"
}
```

Then restart Docker.

75. How do you limit container network bandwidth?

Answer:
Use the `--net` option with traffic control (`tc`) or use Docker networking plugins

for bandwidth control.

76. How do you run multiple containers using Docker Compose?

Answer:
Define services in a `docker-compose.yml` file and use:

```
docker-compose up
```

77. What is the purpose of `HEALTHCHECK` in a Dockerfile?

Answer:
The `HEALTHCHECK` instruction defines a command to test the health of the container, allowing Docker to monitor the application status.

78. What is a Docker context?

Answer:
Docker context allows you to switch between multiple Docker environments (e.g., local, remote hosts).

79. What are Docker labels?

Answer:
Docker labels are metadata applied to Docker objects, such as images and containers, for organizing and filtering resources.

80. How do you build an image without using cache?

Answer:
Use the `--no-cache` option:

```
docker build --no-cache -t <image_name> .
```

81. How do you are data between containers?

Answer:
Use Docker volumes or bind mounts to are data between containers.

82. What is a dangling volume in Docker?

Answer:
A dangling volume is a volume not associated with any container.

83. What is the difference between `docker attach` and `docker exec`?

Answer:

- **`docker attach`: Connects to a container's main process.**
- **`docker exec`: Runs a new process inside an existing container.**

84. What is Docker Stack?

Answer:
Docker Stack is a feature in Docker Swarm to deploy and manage a group of services defined in a YAML file.

85. How do you configure a Docker Swarm manager?

Answer:
Initialize a Swarm:

```
docker swarm init
```

86. What is a Docker node?

Answer:
A node is a machine that is part of a Docker Swarm, either a manager or worker node.

87. How do you promote a Docker node to a manager?

Answer:
Use the command:

```
docker node promote <node_name>
```

88. How do you scale down a Docker Swarm service?

Answer:
Use the command:

```
docker service scale
<service_name>=<number_of_replicas>
```

89. How do you force remove a volume?

Answer:
Use the command:

```
docker volume rm -f <volume_name>
```

90. How do you troubleoot Docker networking issues?

Answer:

- **Use `docker network inspect`.**

- Check container IPs using `docker inspect`.
- Test connectivity using `ping` or `curl`.

91. How do you run a container with privileged access?

Answer:
Use the `--privileged` flag:

```
docker run --privileged <image_name>
```

92. How do you enable live restore in Docker?

Answer:
Configure `/etc/docker/daemon.json` with:

json

```
{
  "live-restore": true
}
```

93. How do you run a container in read-only mode?

Answer:
Use the `--read-only` flag:

```
docker run --read-only <image_name>
```

94. How do you define multiple commands in a Dockerfile?

Answer:
Chain them using `&&` or use multiple `RUN` instructions.

95. What is Docker Content Trust (DCT)?

Answer:
DCT ensures image authenticity by enabling signing of images. It's activated using:

```
export DOCKER_CONTENT_TRUST=1
```

96. How do you view Docker logs in real-time?

Answer:
Use:

```
docker logs -f <container_name_or_id>
```

97. How do you remove orphaned containers in Docker Compose?

Answer:
Use:

```
docker-compose up --remove-orphans
```

98. How do you back up a Docker volume?

Answer:
Use:

```
docker run --rm -v my_volume:/data -v $(pwd):/backup
alpine tar czf /backup/backup.tar.gz /data
```

99. What is the difference between `docker image prune` and `docker system prune`?

Answer:

- `docker image prune`: **Removes dangling images.**
- `docker system prune`: **Cleans up unused containers, networks, and images.**

100. What are Docker layers?

Answer:
Docker layers are read-only filesystems created for each instruction in the Dockerfile, stacked to form the final image. Layers improve efficiency by reusing unchanged parts during builds.