Docker Scenario-based Interview Questions

---

1. **Explain the difference between a Docker container and an image.**
    - **Answer:** A Docker image is a read-only template used to create Docker containers. Containers are instances of Docker images that can be executed and run as isolated environments.

2. **How do you create a Docker image?**
    - **Answer:** Using the `docker build` command. For example:
    - `docker build -t my_image .`

3. **Explain the Dockerfile.**
    - **Answer:** Dockerfile is a text file that contains instructions to build a Docker image. It includes commands like `FROM`, `RUN`, `COPY`, `CMD`, etc.

4. **How can you transfer files from your local system to a Docker container?**
    - **Answer:** Using the `docker cp` command. For example:
    - `docker cp local_file.txt container_id:/path/in/container`

5. **How do you list all Docker containers running on your system?**
    - **Answer:** Using the `docker ps` command. For example:
    - `docker ps`

6. **What command would you use to remove all stopped Docker containers?**

    ○ **Answer:** Using the `docker container prune` command. For example: ○ `docker container prune`

7. **Explain Docker networking and how you can connect containers.**
    - **Answer:** Docker networking allows containers to communicate with each other and the outside world. You can connect containers using bridge networks, overlay networks, or user-defined networks.
8. **How do you expose ports in Docker containers?**
    - **Answer:** Using the `-p` or `--publish` flag with the `docker run` command. For example:
    - `docker run -p 8080:80 my_container`
9. **What is Docker Compose, and how is it used?**
    - **Answer:** Docker Compose is a tool for defining and running multi-container Docker applications. It uses a YAML file to configure the application's services and dependencies.
10. **How do you scale Docker containers horizontally?**
    - **Answer:** Using Docker Swarm or Kubernetes to manage multiple instances of a container across multiple nodes.
11. **Explain Docker volumes and their importance.**
    - **Answer:** Docker volumes are persistent data storage mechanisms that exist outside the container's lifecycle. They are crucial for persisting data between container restarts and for sharing data between containers.
12. **How do you mount a host directory as a volume in a Docker container?**
    - **Answer:** Using the `-v` or `--volume` flag with the `docker run` command. For example:
    - `docker run -v /host/directory:/container/directory my_container`
13. **What is Docker Swarm, and how does it work?**
    - **Answer:** Docker Swarm is a native clustering and orchestration tool for Docker containers. It enables you to create a cluster of Docker hosts and deploy services across them.
14. **How do you update a Docker service in Docker Swarm?**
        - **Answer:** Using the `docker service update` command.

For example:

        - `docker service update --image new_image:tag service_name`

15. **Explain Docker secrets and how they are used.**
    - **Answer:** Docker secrets are encrypted pieces of sensitive information, such as passwords or API keys, that are securely stored and made available only to the services that need them.

16. **How do you manage Docker secrets?**
    - **Answer:** Using the `docker secret` command-line interface or Docker Compose to create, inspect, and remove secrets.

17. **Explain the concept of Docker overlay networks.**
    - **Answer:** Docker overlay networks facilitate communication between Docker services running on different Docker nodes in a Swarm cluster. They provide multi-host communication without requiring links between individual containers.

18. **How do you create an overlay network in Docker Swarm?**
    - **Answer:** Using the `docker network create` command with the `--driver overlay` option. For example:
    - `docker network create --driver overlay my_overlay_network`

19. **Explain the concept of Docker logging drivers.**
    - **Answer:** Docker logging drivers determine where container logs are sent and how they are formatted. They allow you to customize log handling based on your application's needs.

20. **How do you configure a logging driver for a Docker container?**
    - **Answer:** Using the `--log-driver` option with the `docker run` command. For example:
    - `docker run --log-driver=syslog my_container`

21. **What is Docker health check, and why is it important?**
    - **Answer:** Docker health check is a feature that allows you to monitor the health of a container and take action based on its status. It helps ensure the reliability and availability of your applications.

22. **How do you define a health check for a Docker container?**
    - **Answer:** Using the `HEALTHCHECK` instruction in the Dockerfile or the `--healthcmd` option with the `docker run` command.

23. **Explain Dockerfile best practices for building efficient images.**
    - **Answer:** Best practices include using multi-stage builds, minimizing the number of layers, using .dockerignore files, and removing unnecessary dependencies.

24. **What is Docker multi-stage build, and how does it work?**
    - **Answer:** Docker multi-stage build is a feature that allows you to create smaller and more efficient Docker images by using multiple build stages in a single Dockerfile.

25. **How do you remove unused Docker images from your system?**
    - **Answer:** Using the `docker image prune` command. For example:
    - `docker image prune`

26. **Explain Docker container orchestration and its benefits.**
    - **Answer:** Docker container orchestration involves managing the deployment, scaling, and operation of containerized applications across a cluster of machines. It provides benefits such as high availability, scalability, and ease of management.

27. **What is Docker registry, and how is it used?**
    - **Answer:** Docker registry is a repository for storing Docker images. It allows users to push and pull images, making it easy to share and distribute containerized applications.

28. **How do you push a Docker image to a registry?**
    - **Answer:** Using the `docker push` command. For example:
    - `docker push registry.example.com/my_image`

29. **What is Docker content trust, and why is it important?**

- **Answer:** Docker content trust is a security feature that allows you to verify the integrity and authenticity of Docker images. It helps prevent the execution of tampered or malicious images.

30. **How do you enable Docker content trust?**

    - **Answer:** By setting the `DOCKER_CONTENT_TRUST` environment variable to `1`. For example:
    - `export DOCKER_CONTENT_TRUST=1`

31. **Explain how you can troubleshoot Docker container performance issues.**

    - **Answer:** Troubleshooting Docker container performance issues involves monitoring resource usage, analyzing logs, inspecting container

configurations, and using diagnostic tools like `docker stats` and `docker inspect`.

32. **What is Docker Engine, and how does it work?**

    - **Answer:** Docker Engine is the core component of Docker that runs and manages Docker containers. It includes a server (daemon), REST API, and command-line interface.

33. **How do you install Docker Engine on Linux?**

    - **Answer:** By following the official Docker documentation for your Linux distribution.

34. **What is Docker Machine, and what is it used for?**

    - **Answer:** Docker Machine is a tool for provisioning and managing Docker hosts. It allows you to create Docker hosts on local machines, virtual machines, or cloud providers.

35. **How do you create a Docker Machine?**

    - **Answer:** Using the `docker-machine create` command. For example:
    - `docker-machine create --driver virtualbox my_machine`

36. **Explain Docker security best practices.**

- **Answer:** Best practices include using trusted base images, minimizing the attack surface, using Docker security features like content trust and user namespaces, and regularly updating Docker components.

37. **What is Docker stack, and how is it used?**

- **Answer:** Docker stack is a tool for deploying and managing multi-service Docker applications using Docker Compose files in Docker Swarm mode.

38. **How do you deploy a Docker stack in Docker Swarm?**

- **Answer:** Using the `docker stack deploy` command. For example:
- `docker stack deploy -c docker-compose.yml my_stack`

39. **Explain the difference between Docker volumes and bind mounts.**

- **Answer:** Docker volumes are managed by Docker and persist data outside the container's lifecycle. Bind mounts link a directory or file on the host machine to a directory in the container and are subject to filesystem permissions.

40. **How do you secure Docker daemon with TLS certificates?**

- **Answer:** By configuring Docker daemon with TLS certificates and enabling TLS communication between Docker client and daemon.

41. **Explain Docker CPU and memory constraints and how to set them.**

- **Answer:** Docker CPU and memory constraints limit the amount of CPU and memory a container can use. You can set them using the `--cpus` and `-memory` options with the `docker run` command.

42. **What is Docker Hub, and how is it used?**

- **Answer:** Docker Hub is a cloud-based registry service provided by Docker. It hosts public and private Docker images, making it easy to share and distribute containerized applications.

43. **How do you pull a Docker image from Docker Hub?**

- **Answer:** Using the `docker pull` command. For example:
- `docker pull my_image:tag`

44. **Explain Docker build caching and how it improves build performance.**

- **Answer:** Docker build caching caches intermediate layers during the image build process. It improves build performance by reusing cached layers for unchanged build steps.

45. **What is Docker image tagging, and why is it important?**
    - **Answer:** Docker image tagging is the process of assigning a label (tag) to a Docker image. It helps identify different versions of an image and manage image versions effectively.

46. **How do you tag a Docker image?**
    - **Answer:** Using the `docker tag` command. For example:
    - `docker tag my_image:latest my_image:new_tag`

47. **Explain Docker restart policies and their significance.**
    - **Answer:** Docker restart policies define the behavior of a container when it exits. They determine whether the container should be automatically restarted and under what conditions.

48. **How do you set a restart policy for a Docker container?**
    - **Answer:** Using the `--restart` option with the `docker run` command. For example:
    - `docker run --restart=always my_container`

49. **What is Docker swarm mode, and how does it differ from standalone Docker?**
    - **Answer:** Docker swarm mode enables native clustering and orchestration of Docker containers across multiple hosts. It provides features like service discovery, load balancing, and rolling updates, which are not available in standalone Docker.

50. **How do you enable Docker swarm mode?**
    - **Answer:** By initializing a swarm with the `docker swarm init` command on a Docker host. For example:

      ```
      docker swarm init
      ```