linkedin.com/in/nvndops

Important Docker Interview Questions

Q1: What is the difference between an Image, Container, and Engine?

A1:

- **Image:** A lightweight, standalone, and executable software package that includes everything needed to run a piece of software, including the code, runtime, libraries, environment variables, and config files.
- **Container:** A running instance of a Docker image. It's a lightweight, isolated environment where the application runs.
- **Engine:** The Docker Engine is the core software that enables containers to run. It's responsible for running and managing containers on a host system.

Q2: What is the difference between the Docker command COPY vs ADD?

A2:

- **COPY:** Copies files or directories from the local filesystem into the Docker image.
- **ADD:** Similar to COPY, but with additional features. It can also copy files from a URL and automatically unpack compressed files (e.g., .tar, .gz).

linkedin.com/in/nvndops

Q3: What is the difference between the Docker command CMD vs RUN?

A3:

- **CMD:** Specifies the default command to run when a container starts. It can be overridden by passing a command when starting a container.
- **RUN:** Executes commands during the build process of the Docker image, creating a new layer in the image.

Q4: How will you reduce the size of a Docker image?

A4:

- Use multi-stage builds.
- Minimize the number of layers.
- Use .dockerignore to exclude unnecessary files.
- Choose a minimal base image.
- Combine RUN statements to reduce the number of layers.

Q5: Why and when should you use Docker?

A5:

- Docker is used for its portability, consistency across environments, isolation, ease of deployment, and efficient resource utilization.

- It is particularly useful in CI/CD pipelines, microservices architecture, and cloud-based applications.

Q6: Explain the Docker components and how they interact with each other.

A6:

- **Docker Client:** The user interface to interact with Docker.
- **Docker Daemon (Engine):** Handles container lifecycle and images.
- **Docker Images:** Templates for creating containers.
- **Docker Containers:** Isolated environments running instances of images.
- **Docker Registry:** Stores Docker images (e.g., Docker Hub).
- **Docker Network:** Enables communication between containers.
- **Docker Volume:** Manages persistent storage.

Q7: Explain the terminology: Docker Compose, Dockerfile, Docker Image, Docker Container.

A7:

- **Docker Compose:** A tool for defining and running multi-container Docker applications using a docker-compose.yml file.
- **Dockerfile:** A text file that contains instructions for building a Docker image.
- **Docker Image:** A snapshot or template used to create containers.

- **Docker Container:** A running instance of a Docker image.

Q8: In what real scenarios have you used Docker?

A8:

- Running a consistent development environment across teams.
- Deploying microservices.
- Automating testing and integration in CI/CD pipelines.
- Isolating and securing applications in multi-tenant environments.

Q9: Docker vs Hypervisor?

A9:

- **Docker:** Uses OS-level virtualization, shares the host OS kernel, and is more lightweight.
- **Hypervisor:** Uses hardware-level virtualization, runs full virtual machines, and is heavier but more isolated.

Q10: What are the advantages and disadvantages of using Docker?

A10:

- **Advantages:** Portability, resource efficiency, faster CI/CD pipelines, isolation.

- **Disadvantages:** Less isolation than VMs, potential security risks, complexity in managing persistent data.

Q11: What is a Docker namespace?

A11:

- Namespaces provide isolation for containers by partitioning the kernel so that one set of processes sees one set of resources while another set of processes sees another set of resources.

Q12: What is a Docker registry?

A12:

- A service that stores Docker images. Public registries like Docker Hub and private registries are available.

Q13: What is an entry point?

A13:

- The ENTRYPOINT instruction in a Dockerfile configures a container to run as an executable.

Q14: How to implement CI/CD in Docker?

A14:

- Use Docker to create isolated and consistent environments for building, testing, and deploying applications. Integrate Docker with CI/CD tools like Jenkins, GitLab CI, or GitHub Actions.

Q15: Will data on the container be lost when the Docker container exits?

A15:

- By default, data inside a container is ephemeral and will be lost when the container exits. To persist data, use Docker volumes or bind mounts.

Q16: What is a Docker swarm?

A16:

- A native clustering and orchestration tool for Docker. It allows you to manage a cluster of Docker hosts as a single virtual host.

Q17: What are the Docker commands for the following:

- **Viewing running containers:** `docker ps`
- **Running a container under a specific name:** `docker run --name my_container image_name`
- **Exporting a Docker image:** `docker save -o my_image.tar image_name`

- **Importing an existing Docker image:** docker load -i my_image.tar
- **Deleting a container:** docker rm container_name
- **Removing all stopped containers, unused networks, build caches, and dangling images:** docker system prune -a

Q18: What are the common Docker practices to reduce the size of Docker images?

A18:

- Use multi-stage builds, minimal base images, combine RUN instructions, and clean up cache or temporary files.

Q19: How do you troubleshoot a Docker container that is not starting?

A19:

- Check logs (docker logs container_name).
- Inspect the container’s status (docker inspect container_name).
- Verify Dockerfile and configurations.
- Ensure the required services are running.

Q20: Can you explain the Docker networking model?

A20:

- Docker uses a layered network model. The default networks are bridge, host, and none. Containers can communicate across networks using Docker networking.

Q21: How do you manage persistent storage in Docker?

A21:

- To manage persistent storage in Docker, you can use the following commands:

1. Create a Docker Volume:

`docker volume create my_volume`

This command creates a named volume that can be used to persist data.

2. Mount a Volume to a Container:

`docker run -d --name my_container -v my_volume:/path/inside/container image_name`

This command runs a container and mounts the `my_volume` to a specific path inside the container.

3. Bind Mount a Directory:

`docker run -d --name my_container -v /host/path:/container/path image_name`

This command runs a container and mounts a directory from the host system to the container.

4. Inspect a Volume:

`docker volume inspect my_volume`

This command provides detailed information about a volume, including its mountpoint.

5. List Docker Volumes:

`docker volume ls`

This command lists all the volumes available on the Docker host.

6. Remove a Docker Volume:

`docker volume rm my_volume`

This command removes a specific volume.

7. Remove Unused Volumes:

`docker volume prune`

This command removes all unused volumes, freeing up space on the Docker host.

These commands help manage persistent storage in Docker by creating, mounting, inspecting, and removing volumes or bind mounts.

Q22: How do you secure a Docker container?

A22:

- Use minimal base images, apply the principle of least privilege, use Docker's security options (like user namespaces), and keep the Docker Engine up-to-date.

Q23: What is Docker overlay networking?

A23:

- A network driver that allows containers running on different Docker hosts to communicate with each other across a distributed network.

Q24: How do you handle environment variables in Docker?

A24:

- Environment variables can be set in a Dockerfile using ENV, or passed during container runtime using the -e flag or --env-file.