LINUXWORLD
Making India Future Ready

#13

Docker Interview Practice Question Set 1

1. Can you explain the difference between containers and virtual machines?

Answer: Containers and virtual machines (VMs) both provide isolation for running applications, but they do so differently. Containers are lightweight, as they share the host system's OS kernel, which makes them faster to start and more resource-efficient. They provide process-level isolation and allow applications to run consistently across different environments.

Virtual machines, on the other hand, run a full operating system on top of virtualized hardware. This allows for complete OS-level isolation, meaning you can run multiple independent OS instances. However, VMs tend to be more resource-heavy and slower to boot since each VM includes its own OS.

2. What role does Docker Engine play in the Docker ecosystem?

Answer: Docker Engine is the backbone of Docker. It's essentially a client-server application consisting of a server-side component (Docker Daemon) that handles container creation, management, and running, and a client-side component (Docker CLI) for interacting with the daemon. The Docker Engine handles key tasks like container lifecycle management, networking, storage, and building images, and it communicates with the Docker registry to pull or push images.

3. How would you define a Docker image?

Answer: A Docker image is essentially a snapshot of a filesystem that includes everything an application needs to run, such as code, runtime, libraries, dependencies, and system tools. These images are immutable and portable, meaning they can run on any machine that supports Docker. Images are created using a Dockerfile, and they serve as the blueprint for running containers.

4. What is Docker Hub, and how does it contribute to the container ecosystem?

Answer: Docker Hub is a cloud-based service for storing and sharing Docker images. It acts as a registry, allowing developers to push and pull Docker images, whether they're public or private. It's a convenient platform to access and distribute pre-built images, either for general use or specific to an organization, making container deployment much easier and faster.

5. Could you walk me through how you'd launch a container from an existing Docker image?

Answer: To launch a container, you simply use the docker run command followed by the image name. For instance, running docker run myimage:tag would create and start a container from the specified image. You can also pass additional flags to configure the container, such as setting environment variables, linking volumes, or specifying network configurations.

6. What is a Dockerfile and how does it relate to Docker images?

Answer: A Dockerfile is a script composed of a series of instructions that define how a Docker image should be built. It includes steps like copying files into the image, setting environment variables, installing dependencies, and defining what commands to run. The Dockerfile allows you to create a custom image tailored for your application by automating the setup process.

7. Can you explain the function of a hypervisor in virtualization?

Answer: A hypervisor is the software layer that enables virtualization. It divides the host machine's resources and allocates them to guest virtual machines. Hypervisors come in two types:

- **Bare-metal (Native) Hypervisors**: These run directly on the host hardware and manage the guest VMs without the need for a host OS.
- **Hosted Hypervisors**: These run on top of an existing operating system and use the host's resources to manage VMs. Hypervisors are essential

- for running multiple operating systems on the same physical machine, providing isolation and resource management for each VM.

8. How would you go about creating a Docker image using a Dockerfile?

Answer: To create a Docker image from a Dockerfile, you would use the docker build command, followed by the -t flag to specify a tag and the path to the directory containing the Dockerfile. For example, docker build -t myimage:latest . would build an image from the current directory. The Docker daemon processes the instructions in the Dockerfile and assembles the image layer by layer.

9. If I needed to start and stop a Docker container, how would I go about doing that?

Answer: To start a container, you would use the docker start command followed by either the container's name or ID. For example, docker start mycontainer will start a container named "mycontainer". To stop a running container, you'd use docker stop mycontainer, which gracefully stops the container.

10. Can you list and briefly describe the key components of Docker?

Answer: Docker has three main components:

- **Docker Client**: This is the interface you use to interact with Docker. It sends requests to the Docker Daemon via the Docker API.
- **Docker Daemon (Docker Host)**: The core of Docker, it runs the containers, manages images, and handles networking, storage, and other essential functions.
- **Docker Registry**: This is where Docker images are stored. Docker Hub is the default public registry, but private registries can also be used. Images are pulled from the registry to run containers.

11. What command do you use to delete a Docker container?

Answer: To remove a Docker container, use the docker rm <container_name> or docker rm <container_id> command. However, if the container is still running, you must stop it first with docker stop <container_name> before removing it.

12. How can you clean up stopped containers and unused networks in Docker efficiently?

Answer: Docker provides the docker system prune command, which helps remove stopped containers, unused networks, dangling images, and build caches in one go. This is a great way to free up system resources and keep your Docker environment clean.

13. Will you lose data if a Docker container stops or exits?

Answer: No, stopping or exiting a container does not erase data stored inside it. However, if the container is removed (docker rm), any non-persistent data will be lost unless you use volumes or bind mounts to store data externally.

14. Is there a limit on the number of containers you can run in Docker?

Answer: There is no hard limit on the number of containers Docker can run. However, the actual number depends on system resources such as CPU, memory, and disk space. The more powerful the hardware, the more containers you can efficiently run.

15. What is the purpose of the FROM directive in a Dockerfile?

Answer: The FROM directive in a Dockerfile specifies the base image to use for building the Docker image. It is the first instruction in a Dockerfile, and all subsequent instructions are executed on top of this base image. The base image could be a lightweight image like alpine or a full-fledged operating system like ubuntu.

16. How would you use the COPY instruction in a Dockerfile?

Answer: The COPY instruction is used to copy files and directories from the host machine into the Docker image. It takes two arguments: the source file/directory on the host and the destination path inside the image. Example: COPY ./app /usr/src/app would copy the app folder from the local directory to /usr/src/app in the image.

17. What is the difference between COPY and ADD in a Dockerfile?

Answer: Both COPY and ADD are used to copy files into a Docker image, but ADD has more capabilities. In addition to copying files like COPY, ADD can also extract tar archives automatically and support remote URLs. However, it's generally recommended to use COPY unless these extra features of ADD are needed, as COPY is simpler and more predictable.

18. What is the role of the RUN instruction in a Dockerfile?

Answer: The RUN instruction in a Dockerfile is used to execute commands inside the image during the build process. It typically installs packages or performs setup tasks. Each RUN command creates a new layer in the image, so it's best to combine multiple commands in a single RUN statement to reduce the image size.

19. What is the purpose of the ENTRYPOINT directive in a Dockerfile?

Answer: The ENTRYPOINT directive in a Dockerfile defines the default executable that will run when a container is started from the image. Unlike CMD, which can be overridden at runtime, ENTRYPOINT sets the executable and allows arguments to be passed to it. Example: ENTRYPOINT ["python", "app.py"].

20. How does the CMD instruction work in a Dockerfile?
Answer: The CMD instruction in a Dockerfile provides default arguments for the ENTRYPOINT or specifies the command to run when the container starts. Unlike ENTRYPOINT, CMD can be overridden at runtime. You can use CMD to specify things like the default command or the command with arguments.