Interview Questions :

1. What is Virtualization and Containerization?

- Virtualization is a technology that allows multiple virtual machines (VMs) to run on a single physical server, each operating as an isolated and independent environment.
- It abstracts the hardware resources and enables efficient utilization of server resources by running multiple VMs with different operating systems on the same host.

- Containerization, on the other hand, is a lightweight form of virtualization that allows multiple containers to run on a single host, sharing the host's kernel but isolated from each other.
- Containers package an application and its dependencies together, providing a portable and consistent runtime environment across different environments.

2. What is Containerization?

- Containerization is a technology that enables the packaging of an application and its dependencies into a single container.
- Containers provide an isolated and lightweight runtime environment, ensuring that the application runs consistently across various platforms, from development to production.

- Each container shares the host's kernel, making them faster to start, use fewer resources, and offer higher performance compared to traditional virtual machines.
- Containerization allows developers to build, test, and deploy applications easily and consistently across different environments, making it an essential component of modern DevOps practices.

3. Explain Docker Architecture.

- Docker is a popular containerization platform that simplifies the process of creating, deploying, and running applications within containers.
- Its architecture consists of three main components:

a. Docker Engine:

- The core component responsible for managing containers.
- It includes a server that listens for Docker API requests and a daemon that runs and manages containers on the host system.

b. Docker Images:

- These are read-only templates used to create containers.
- Images contain the application code, libraries, and dependencies required to run the application.

- Images are built from Dockerfiles and can be shared and distributed through Docker Hub or other container registries.

c. Docker Containers:

- These are instances of Docker images. Containers are lightweight, isolated, and portable, running applications in an environment with their own file systems, processes, and networking.
- They can be started, stopped, and managed independently of other containers.

- Docker's architecture simplifies the deployment and scaling of applications, making it a popular choice for containerization in modern software development and cloud-based infrastructure.

4. What is Docker Hub?

- Docker Hub is a cloud-based registry service provided by Docker that allows developers to store, share, and manage Docker container images.
- It serves as a centralized repository for Docker images, making it easy to find and pull images to run containers on various environments.
- Docker Hub also offers collaboration features, allowing teams to share private images and automate the build and deployment processes using Docker automated builds.

5. How do you Install Docker?

- Docker can be installed on various operating systems, including Linux, macOS, and Windows.
- The installation steps vary based on the OS. Here are the general steps for installing Docker on Linux:

a. Update package index: Run sudo apt-get update (for Debian/Ubuntu) or sudo yum update (for CentOS/RHEL).

b. Install required packages: Run sudo apt-get install docker.io (for Debian/Ubuntu) or sudo yum install docker (for CentOS/RHEL).

c. Start Docker service: Run sudo systemctl start docker.

d. Enable Docker on system boot: Run sudo systemctl enable docker.

e. Verify Docker installation: Run docker --version to check the installed Docker version.

6. What are some commonly used Docker commands?

- Docker provides a rich set of commands to manage containers and images.
- Some commonly used Docker commands are:

- docker pull <image>: Pulls a Docker image from a registry (e.g., Docker Hub).
- docker run <image>: Creates and starts a new container from a specified image.
- docker ps: Lists all running containers.
- docker ps -a: Lists all containers (both running and stopped).
- docker stop <container>: Stops a running container.
- docker start <container>: Starts a stopped container.
- docker rm <container>: Removes a stopped container.
- docker images: Lists all downloaded Docker images.
- docker rmi <image>: Removes a Docker image from the local system.
- docker logs <container>: Displays the logs of a container.
- docker exec -it <container> <command>: Executes a command inside a running container.
- These commands form the basis of interacting with Docker and managing containers and images effectively.

7. What are the Container Modes in Docker?

- Docker supports two container modes: "attached" and "detached."

a. Attached Mode:

- In attached mode (the default), the container's standard input (stdin), standard output (stdout), and

standard error (stderr) are directly connected to the terminal from which the container is started.
- The container's output is visible in real-time, and the container process runs in the foreground.

b. Detached Mode:
- In detached mode, also known as background mode, the container runs as a background process, and its input/output is detached from the terminal.
- The container continues running even if the terminal is closed, and you can later attach to the container's output using the docker attach command.

8. What is Port Binding in Docker?
- Port binding is a mechanism that allows you to map a port from the host machine to a port inside the Docker container.
- It enables communication between the container and the host or other containers.
- Port binding is essential when running services inside containers that need to be accessible from the outside world.

- For example, to bind port 8080 on the host to port 80 inside the container, you would use the following command:

```
docker run -p 8080:80 <image-name>
```

9. What is Docker Volume?

- A Docker volume is a persistent data storage mechanism that allows data to be stored outside the container and shared between containers.
- Volumes ensure that data is preserved even when containers are stopped or removed.
- They provide a way to manage data persistence independently of the container lifecycle.

- Volumes can be mounted into containers using the -v or --volume option when using the docker run command. For example, to create and mount a volume named "mydata" inside a container, you would use:

```
docker run -v mydata:/app/data <image-name>
```

10. What is Docker Compose?

- Docker Compose is a tool for defining and running multi-container Docker applications.
- It uses a YAML file (docker-compose.yml) to define the services, networks, and volumes required for an application.
- With Docker Compose, you can start, stop, and manage all the services of your application with a single command.

- Docker Compose simplifies the management of complex multi-container applications, making it easier to set up and tear down environments consistently.

- It is a valuable tool for defining the infrastructure and services required for modern microservices architectures.