Docker architecture

- Docker architecture relies on a client-server model where the Docker client interacts with the Docker daemon to manage containers. The Docker client uses a command-line interface (CLI) or other applications to send commands to the daemon, which then builds, runs, and manages containers. The daemon also manages images, networks, and volumes.

Docker Client:

- The interface through which users interact with Docker, sending commands like docker build, docker run, and docker stop to the daemon.

Docker Daemon:

- A background process on the Docker host that listens for API requests from the client and manages Docker objects (images, containers, networks, volumes).

Docker Host:

- The physical or virtual machine running the Docker daemon, where containers are built and executed.

Docker Registry:

- A repository where Docker images are stored and can be pushed to and pulled from. Docker Hub is a public registry, but users can also run their own.

Docker Images:

- Read-only templates that define the application's environment, including code, runtime, system tools, libraries, and settings.

Docker Containers:

- Instances of Docker images that are running applications, isolated from each other and the host operating system.

Docker API:

- A set of APIs that allows clients to interact with the daemon, often exposed over a REST API.
- In essence, Docker leverages a client-server architecture to provide a consistent environment for building, running, and deploying applications across different environments. This allows for portability, isolation, and scalability, making it a valuable tool for developers and system administrators.