**Introducing Docker, What it does and why to use it?**

Docker is an open-source platform that provides a way to automate the deployment, scaling, and management of applications using containerization. Containerization allows you to package an application and its dependencies into a standardized unit called a container.

Here are a few reasons why you might want to use Docker:

1. \*\*Consistent and reproducible environments:\*\* With Docker, you can create a container that includes all the dependencies and configurations needed to run your application. This ensures that your application will run consistently, regardless of the environment it is deployed in or the host operating system.

2. \*\*Efficient resource utilization:\*\* Containers are lightweight and share the host OS kernel, which means they require fewer resources compared to running each application on a separate virtual machine. This allows you to maximize the utilization of your infrastructure resources.

3. \*\*Isolation and security:\*\* Docker containers provide a level of isolation between applications, preventing one application from impacting others. Additionally, Docker provides various security features to help protect your applications and data.

4. \*\*Fast and easy deployment:\*\* Docker simplifies the deployment process by encapsulating your application and its dependencies into a container. This makes it easy to deploy your application on any host that has Docker installed, with minimal setup and configuration required.

5. \*\*Scalability:\*\* Docker makes it easy to scale your application horizontally by running multiple instances of your container across a cluster of machines. This allows you to handle increased traffic or workload by simply adding more containers.

Overall, Docker provides a consistent and efficient way to package, distribute, and run applications, making it easier to deploy and manage your software across different environments.

**Docker images, Docker CLI**

Docker images are the building blocks of containers in the Docker ecosystem. They are essentially lightweight, standalone, and executable software packages that bundle everything needed to run an application, including the code, runtime, libraries, and system tools.

Docker CLI (Command Line Interface) is the command-line tool that allows you to interact with Docker and perform various operations related to containers and Docker images. It provides a set of commands that you can use to create, manage, and deploy containers.

Here are some key points about Docker images and the Docker CLI:

1. Docker Images:

- Docker images are created based on a Dockerfile, which is a text file that contains instructions for building the image.

- Images are stored in a registry, such as Docker Hub, where you can find pre-built images or upload your own.

- Images can be layered, meaning that you can have a base image and then add additional layers on top of it.

- Docker images are immutable, meaning they cannot be changed once created. Any modifications result in the creation of a new image.

2. Docker CLI:

- Docker CLI allows you to manage containers, images, networks, volumes, and other Docker components.

- The CLI provides a set of commands, such as `docker run`, `docker build`, `docker pull`, and `docker push`, to perform various operations.

- You can use the CLI to start, stop, restart, and remove containers, as well as check their logs and status.

- The CLI also provides options to manage networks and volumes, which are used for communication between containers and data persistence.

To work with Docker, you would typically start by pulling or building a Docker image using the CLI. Then, you can use the image to create and manage containers as per your application's requirements. The CLI provides a simple and consistent interface for managing Docker resources and is widely used for containerization and deployment purposes.

Docker is a very popular and powerful open-source containerization platform that is used for building, deploying, and running applications. Docker allows you to decouple the application/software from the underlying infrastructure.  
  
**What is a Container?**  
A container is a standard unit of software bundled with dependencies so that applications can be deployed fast and reliably b/w different computing platforms.

* Docker can be visualized as a big ship (docker) carrying huge boxes of products (containers).
* Docker container doesn’t require the installation of a separate operating system. Docker just relies or makes use of the kernel’s resources and its functionality to allocate them for the CPU and memory it relies on the kernel’s functionality and uses resource isolation for CPU and memory, and separate namespaces to isolate the application’s view of the OS (operating system).

**Why Learn Docker?**  
Application development is a lot more than just writing code! They involve a lot of behind-the-scenes things like usage of multiple frameworks and architectures for every stage of its lifecycle which makes the process more complex and challenging. Using the nature of containerization helps developers to simplify and efficiently accelerate the application workflow along with giving them the liberty to develop using their own choice of technology and development environments.

* All these aspects form the core part of DevOps which becomes all the more important for any developer to know these in order to improve productivity, fasten the development along with keeping in mind the factors of application scalability and more efficient resource management.
* Imagine containers as a very lightweight pre-installed box with all the packages, dependencies, software required by your application, just deploy to production with minimal configuration changes.
* Lots of companies like PayPal, Spotify, Uber, etc use Docker to simplify the operations and to bring the infrastructure and security closer to make more secure applications.
* Being portable, Containers can be deployed on multiple platforms like bare instances, virtual machines, Kubernetes platform etc. as per requirements of scale or desired platform.