**Docker Overview**

As most organizations use DevOps in software/application development, understanding DevOps implementation is very important.

DevOps demands building, testing & deploying the applications in the fastest way possible. Generally, the Development team (Dev team) writes, analyzes, compiles & builds the code. The Operation team (Ops team) tests the code and finally deploys it to production using multiple tools. The same process is implemented in the DevOps life cycle.

Previously before DevOps came into the picture, there was always a gap between the Dev team and the Ops team regarding the built code and expected code output in deployment. The necessity for both the development team and the Operation team to use a single common platform for completing all the processes like building the code, running the code, and testing the code in the DevOps cycle has become essential.

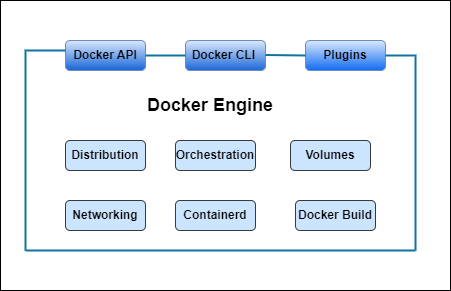
Docker or Containerization technology seems to be one of the best solutions for implementing all these processes.

**2.1 Introduction to Docker:**

Docker Container technology was launched in 2013 as an Open Source Docker Engine. It is open source and can be downloaded and installed for free. Tasks like creating a container, building an image, and deploying an application are also free and easy. In the later years of its release, they made docker into two versions – Product Version (licensed) & Community Version (Non-Licensed).

The Docker Platform provides multiple tools

1. **Docker Engine**
2. Docker Engine is available in two editions - Community Edition which is an open source that can be downloaded and installed for free and Enterprise Edition where the License key is purchased and you get support from the docker community. Docker Engine is the tool used to create & run containers. Docker Engine is used to build, spin and share containers. Docker Engine (DIY) means Do It Yourself i.e., if you are using docker engine then you only have to maintain, configure, update security patches, and monitor images from security vulnerabilities of containers.
3. As Docker Engine runs on different platforms it is called **cross-platform**. It runs on different platforms like Virtual Machines (created in Oracle Virtual Box); Bare metal Servers (directly install Docker Engine/Docker Desktop on it), Cloud (VMs created in any Cloud), Windows/ Linux/MAC OS (install desktop version).
4. Since Docker Engine is platform-independent, container migration from one platform to another platform is possible i.e., the container created on the Windows platform using Docker Engine can be migrated to the Linux platform. Here the container is being migrated, not the OS.
5. Small-scale applications like Nginx, Tomcat, MYSQL DB, and Apache web servers and large-scale applications like Hadoop, Big data can be run in Containers. Applications developed in any language can be run on Docker Engine.
6. Docker Engine is used for Orchestration, Distribution, used for containerd, building images, creating volumes, use networking, and also integrating third-party tools using plugins, run Docker CLI & API calls.



1. **Docker Desktop**

If you want all the features of docker to be available in one single platform/tool then Docker Desktop is the best. The features like running Kubernetes PODs, Security runs are fast, and Patching and Bug fixing are done automatically in Docker Desktop.

If these are to be done by any individual then Docker Engine is used.

1. **Docker Compose**

Docker Compose is used for Orchestration which is nothing but writing a file and giving it to Docker Engine. In Docker Compose, a file with all the parameters such as what type of container to be created, image to take, Volumes, and Port forwarding are written and this file is given as input to the Docker Engine which creates the container.

1. **Docker Swarm**

Docker swarm was used for cluster creation before Kubernetes. As the contribution to Docker swarm decreased and Kubernetes started providing advanced features, docker swarm became less popular and used less.