Docker is a centralized platform for packaging, deploying, and running applications. Before Docker, many users face the problem that a particular code is running in the developer's system but not in the user's system. So, the main reason to develop docker is to help developers to develop applications easily, ship them into containers, and can be deployed anywhere.

Docker was firstly released in March 2013. It is used in the Deployment stage of the software development life cycle that's why it can efficiently resolve issues related to the application deployment.

What is Docker?

Docker is an **open-source centralized platform designed** to create, deploy, and run applications. Docker uses **container** on the host's operating system to run applications. It allows applications to use the same **Linux kernel** as a system on the host computer, rather than creating a whole virtual operating system. Containers ensure that our application works in any environment like development, test, or production.

Docker Containers

Docker containers are the **lightweight** alternatives of the virtual machine. It allows developers to package up the application with all its libraries and dependencies, and ship it as a single package. The advantage of using a docker container is that you don't need to allocate any RAM and disk space for the applications. It automatically generates storage and space according to the application requirement.

Virtual Machine

A virtual machine is a software that allows us to install and use other operating systems (Windows, Linux, and Debian) simultaneously on our machine. The operating system in which virtual machine runs are called virtualized operating systems. These virtualized operating systems can run programs and preforms tasks that we perform in a real operating system.

Advantages of Docker

There are the following advantages of Docker -

- It runs the container in seconds instead of minutes.
- It uses less memory.
- It provides lightweight virtualization.
- It does not a require full operating system to run applications.
- It uses application dependencies to reduce the risk.
- Docker allows you to use a remote repository to share your container with others.
- It provides continuous deployment and testing environment.

Disadvantages of Docker

There are the following disadvantages of Docker -

- It increases complexity due to an additional layer.
- In Docker, it is difficult to manage large amount of containers.
- Some features such as container self -registration, containers self-inspects, copying files form host to the container, and more are missing in the Docker.
- Docker is not a good solution for applications that require rich graphical interface.
- Docker provides cross-platform compatibility means if an application is designed to run in a Docker container on Windows, then it can't run on Linux or vice versa.

Docker Container and Image

Docker container is a running instance of an image. You can use Command Line Interface (CLI) commands to run, start, stop, move, or delete a container. You can also provide configuration for the network and environment variables. Docker container is an isolated and secure application platform, but it can share and access to resources running in a different host or container.

An image is a read-only template with instructions for creating a Docker container. A docker image is described in text file called a **Dockerfile**. This has a simple, well-defined syntax. An image does not have states and never changes. Docker Engine provides the core Docker technology that enables images and containers.

You can understand container and image with the help of the following command.

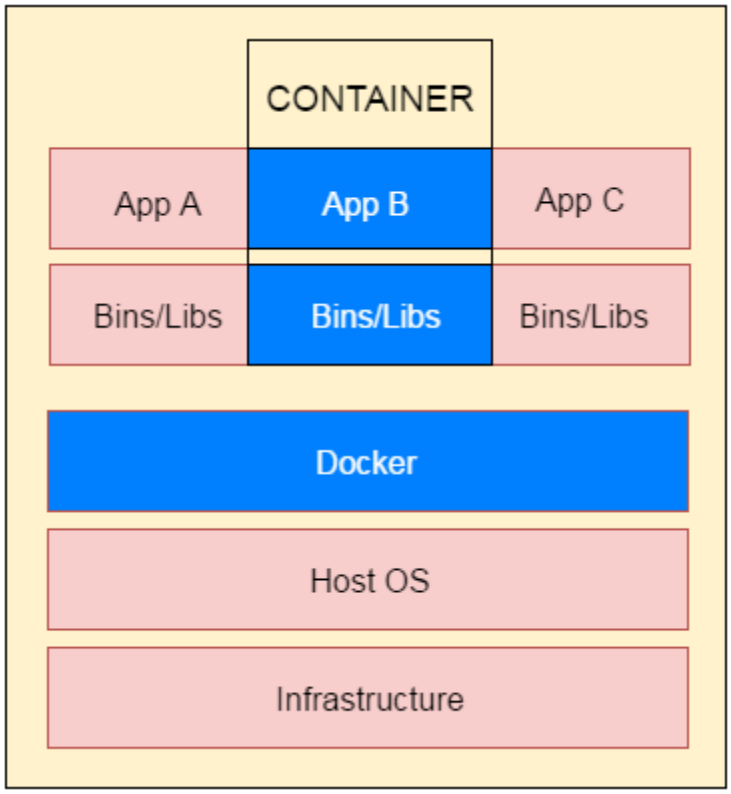
$ docker run hello-world

The above command **docker run hello-world** has three parts.

1) **docker:** It is docker engine and used to run docker program. It tells to the operating system that you are running docker program.

2) **run:** This subcommand is used to create and run a docker container.

3) **hello-world:** It is a name of an image. You need to specify the name of an image which is to load into the container.



Docker Dockerfile

A Dockerfile is a text document that contains commands that are used to assemble an image. We can use any command that call on the command line. Docker builds images automatically by reading the instructions from the Dockerfile.

The docker build command is used to build an image from the Dockerfile. You can use the -f flag with docker build to point to a Dockerfile anywhere in your file system.

Dockerfile Instructions

The instructions are not case-sensitive but you must follow conventions which recommend to use uppercase.

Docker runs instructions of Dockerfile in top to bottom order. The first instruction must be FROM in order to specify the Base Image.

. A statement begins with # treated as a comment. You can use RUN, CMD, FROM, EXPOSE, ENV etc instructions in your Dockerfile.

Here, we are listing some commonly used instructions.

FROM

This instruction is used to set the Base Image for the subsequent instructions. A valid Dockerfile must have FROM as its first instruction.

Ex.
FROM ubuntu

COPY

This instruction is used to copy new files or directories from source to the filesystem of the container at the destination.

Ex.

COPY abc/ /xyz

- ✓ The source path must be inside the context of the build. We cannot COPY ../something /something because the first step of a docker build is to send the context directory (and subdirectories) to the docker daemon.
- ✓ If source is a directory, the entire contents of the directory are copied including file system metadata.

WORKDIR

The WORKDIR is used to set the working directory for any RUN, CMD and COPY instruction that follow it in the Dockerfile. If work directory does not exist, it will be created by default.

We can use WORKDIR multiple times in a Dockerfile.

Ex

WORKDIR /var/www/html

Docker Java Application Example

As, we have mentioned earlier that docker can execute any application.

Here, we are creating a Java application and running by using the docker. This example includes the following steps.

1. Create a directory

   Directory is required to organize files. Create a director by using the following command.

   ```
   $ mkdir  java-docker-app
   ```

2. Create a Java File

   Now create a Java file. Save this file as Hello.java file.

   ```
   // Hello.java
   ```

```
1. class Hello{
2. public static void main(String[ ] args) {
3. System.out.println("This is java app \n by using Docker");
4. }
5. }
```

   Save it inside the directory **java-docker-app** as Hello.java.

3. Create a Dockerfile

   After creating a Java file, we need to create a Dockerfile which contains instructions for the Docker. Dockerfile does not contain any file extension. So, save it simple with **Dockerfile** name.

// Dockerfile

FROM java:8

1. COPY. /var/www/java
2. WORKDIR /var/www/java
3. RUN javac Hello.java
4. CMD ["java", "Hello"]

Write all instructions in uppercase because it is convention. Put this file inside **java-docker-app** directory. Now we have Dockerfile parallel to Hello.java inside the **java-docker-app** directory.

See, your folder inside must look like the below.

4. Build Docker Image

After creating Dockerfile, we are changing working directory.

```
$ cd   java-docker-app
```

Now, create an image by following the below command. we must login as root in order to create an image. In this example, we have switched to as a root user. In the following command, **java-app** is name of the image. We can have any name for our docker image.

```
$ docker build -t java-app .
```

After successfully building the image. Now, we can run our docker image.

5. Run Docker Image

After creating image successfully. Now we can run docker by using run command. The following command is used to run java-app.

```
$ docker run java
```

Now, we have run docker image successfully.