**Docker**

There are various life cycle of software development:

1. Design
2. Development
3. Deployment
4. Testing

Docker makes the process of application deployment very easy and efficient and resolves a lot of issues related to deploying application.

Docker is the world’s leading software container platform.

So a developer will package all of the software’s components, libraries into simple **CONTAINER**. Docker will take care for shipping this container to all the platform in a standard way.

So, now developer should only concern about creating the code and the software and willpackage the software along with all its dependencies and libraries and not worry about how it is deployed on what al platform.

**How it works?**

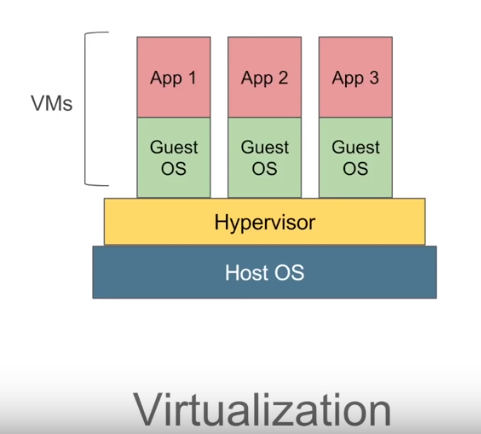
In general scenario, developer will define all the application and its dependencies in a file called **Dockerfile**. This Dockerfile will be used to create the docker image. So, it Docker image all the allpication and its dependencies id present. When you run the docker image you get docker containers. Docker containers are the runtime instances of the Docker image.

These Docker images can also be stored in the online cloud repository which is called Docker Hub.

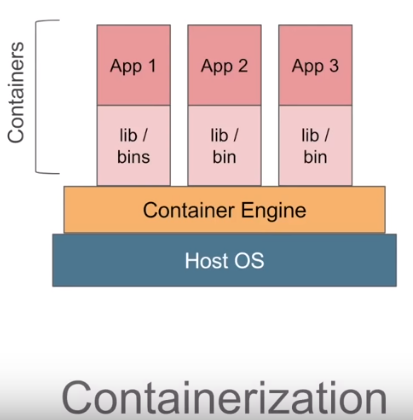
These images can be pulled to create containers in any environment.

**Containerization VS Virtualization**

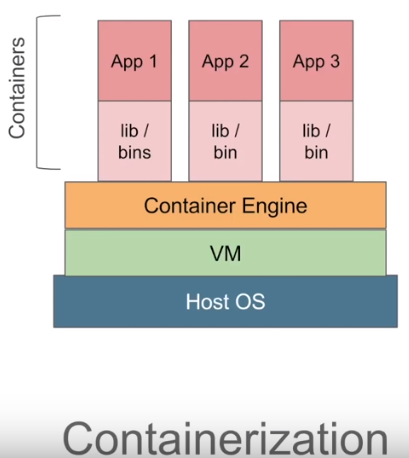
In virtualizatiuon, we have a software called Hypervisor which we used in our Host OS. Hypervisor is used to create and run virtual machines. Using Hypervisor, we can create multiple VM’s on the host OS. These virtual machine have their own OS i.e, it does not uses the host operating system. So, there can be a overhead on host platform. Also we have to allocate fixed memory for every VM’s so there can be a wastage of lot of memory and space.



In containerization, we have a conatainer engine and we do not have separate OS but we have container which have application and all its dependencies. It will use the hist operating system unlike Virtualization which uses its own OS. Here, memory, space, other resources are not fixed i.e, they are taken dynamically so t is very fast and lightweight.



There can be a scenario where we need VM over our host OS and then have container over VM. For example, if you want to use windows operating system over linux OS we need to have VM first which will have a windows OS and we can have a containers over it.



Docker has a client server architecture,

.In Docker, Command line interface is a client and we have a Docker server or a Docker Daemon which will have all the containers. Docker server receives the commands from the Docker client in the form of commands or a rest API request. Docker client with Docker server together form a ***DOCKER ENGINE***.

Docker Client and Daemon can be present in the same or different HOST(machine).

**Advantages:**

It resolves a problem of a code working on one system and not working on different system. So, with Docker you can build your application only once and then there is no need to build or configure multiple times on different encironments of platforms.

Suppose we have an application on Java. We build it and using maven, Dockerfile we create an image which will contain that jar/war, jvm and all the environment in which it is running.

So, In short docker image will contain that jar/war and all the environment setup.

We now can hand this image to anyone and the jar/war will not give the trouble to run as docker image will have all the environment set ip already.

We run this image which will lead to the creation of the instance of that image which is called as container.

Container does not contain the full OS, and that’s why it it very light weight and much more efficient than running VM’s.

Suppose we have Linux OS. Containers are kind of processes that runs on that Linux Kernal. All the containers shares the same Linux Kernal that is why containers are different than VM’s as each VM has its own kernal.

Instead of building war/jar file and handing it over for the deployment, we built a Docker Image. Then its easier for the deployer as they just run the container on that hardware.

These containers are nothing new, as they are there for a very long time as they are the features of Linux kernal since 2008.

Following steps is done by docker :-

1. The Docker client contacts the Docker daemon.
2. The Docker daemon pulls image from the Docker Hub.
3. The Docker daemon creates a new container from that image which runs the executable that produces the output.
4. The Docker daemon streamed that output to the Docker client, which sent it to your terminal.

**Docker Toolbox VS Docker Windows**

**Docker Toolbox:** In older version of windows, you need to have Docker Toolbox. The difference is that in the background it will install a copy of Oracle Virtual box and runs Linux kernal in a Virtual Machine(VM) all in the background.

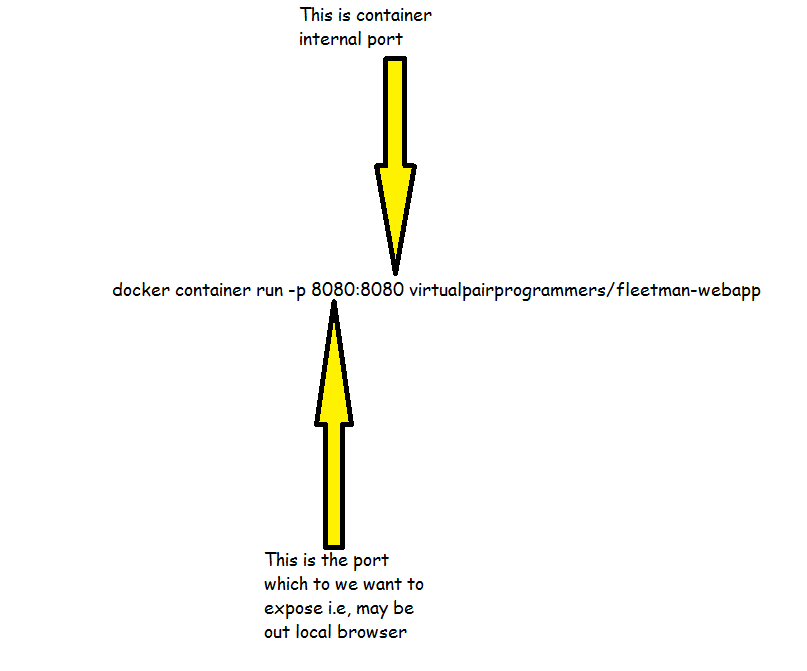
**Docker or Windows/Mac:** In this also Linux virtual machine is running, but you are using the native virtualization support from your HOST OS(Hyper-V on windows). So, it does not have to run Oracle Virtual box.

Commands

1. docker image pull <IMAGE NAME>

* download an image from DockerHub

1. docker image ls
2. docker-machine ip



To install jdk in the container, we use interactive bash of link:-

$ docker container run –it ubuntu

root@12332:/# apt-get install openjdk-8

root@12332:/# javac

root@12332:/# exit

$ docker container commit -a "tanuj tripathi tanuj.tiwari99@gmail.com" <container-Id> myjdkubuntuimage

In the above we are making use of the ubuntu image which is available publicly and adding jdk in it.

If we want to do it through Dockerfile:-

FROM ubuntu:latest

MAINTAINER Tanuj Tripathi "tanuj tripathi [tanuj.tiwari99@gmail.com](mailto:tanuj.tiwari99@gmail.com)"

RUN apt-get update && apt-get install –y openjdk-8-jdk

CMD [“/bin/bash”]