Docker File :

Docker can build images automatically by reading the instructions from a Dockerfile. A Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image. Using docker build users can create an automated build that executes several command-line instructions in succession.

Creating file named Dockerfile :

```
#mkdir /images/DockerFiles
#touch /images/DOckerFiles/Dockerfile
#vim /images/DOckerFiles/Dockerfile
FROM centos:centos7
MAINTAINER Mahesh mahesh@xyz.com
RUN yum install java -y
RUN mkdir /opt/tomcat
WORKDIR /opt/tomcat
ADD https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.73/bin/apache-tomcat-9.0.73.tar.gz .
RUN tar -xvzf apache-tomcat-9.0.73.tar.gz
RUN mv apache-tomcat-9.0.73/* /opt/tomcat
ADD https://github.com/write4mahesh/javapp/blob/master/addressbook.war /opt/tomcat/webapps/
EXPOSE 8080
CMD ["/opt/tomcat/bin/catalina.sh", "run"]
:wq!
```

Each instruction creates one layer:

FROM : creates a layer from the ubuntu:18.04 Docker image.
MAINTAINER : docker FIle Author Details
COPY : adds files from your Docker client's current directory.
RUN : builds your application with make.
CMD : specifies what command to run within the container.

#docker build /images/DockerFiles **(Creating docker image from the docker File with out specifying name & Tag)**

#docker build -t image1:1.0 /images/DockerFiles **(Creating docke>r image from docker file with required name and Tag)**

Running the Image :
#docker images
#docker run <imageid>

Image Layers :

When you pull a Docker image, you will notice that it is pulled as different layers. Also, when you create your own Docker image, several layers are created. we will try to get a better understanding of Docker layers.

A Docker image consists of several layers. Each layer corresponds to certain instructions in your Dockerfile. The following instructions create a layer: RUN, COPY, ADD. The other instructions will create intermediate layers and do not influence the size of your image.

#docker image ls
#docker history nginx **(Displays the layers of changes made in the Image)**

Image Tagging & Pushing to docker hub :
#docker pull nginx
#docker pull nginx:mainline
#docker image ls

NOTE :
It is actually already knew that based on the imageid, This image already exists in cache.Thats the reason in the output both images of nginx are showing the same Image ID bcz they are not really stored twice in the cache.

How to make new Labels :

#docker image tag nginx useridofdockerhub/nginx **(Since we have not given any tag it takes latest tag name)**

#docker images ls

Lets upload this image to docker hub :

#docker login
#docker image push useridofdockerhub/ninx
#docker image tag useridofdockerhub/nginx useridofdockerhub/nginx:testing
#docker image ls
#docker image push useridofdockerhub/nginx:testing

NOTE : **Go to www.hub.docker.com , Login as user and verify the pushed images in the repo.**