Docker Handson

Question: How to build the docker image ?

Tasks to do: Use OSlib given in docker hub (ubuntu:18.04)
On top of it
Add your own java version
Add your own tomcat software bundle
Deploy the war to it

To build the docker image use the below command :

Most important if this is the fresh configuration need to install the Docker in it..
Use the below link to install:

https://github.com/lerndevops/labs/tree/master/docker/install

Step 1:Install the ubuntu image

docker pull ubuntu:18.04

Step2: Check it is installed or not
docker image ls

Step3:we are downloading some softwarebundles with the below link with command

wget https://github.com/lerndevops/code/blob/main/sampleapp.war

```
$ sudo wget
https://archive.apache.org/dist/tomcat/tomcat-9/v9.0.21/bin/apache-tomcat-
9.0.21.tar.gz
```

wget https://github.com/lerndevops/code/raw/main/jdk-11-0-17-linux-x64.tar.gz

Now we have 3bundles in the local which we want to convert into a single docker image

Step4:Now unzip the file
tar -xzf apache-tomcat-9.0.21.tar.gz
tar -xzf jdk-11-0-17-linux-x64.tar.gz
remove the tar files by rm command

Step5:This is show the interactive cli to check the files inside it

docker run -it ubuntu:18.04 /bin/bash

Step6:Create the Docker container as till now we have pull one image that is ubuntu

docker container run -d nginx:latest

Step7:To check how many container are running

docker container ps -a

Step8:Here i'm setting the goble java environment "/opt/java" is the path "export" is linux command "JAVA_HOME" keyword

export JAVA_HOME=/opt/java

Step9: To check if the above command is working or not by "env" and we got this after see the
SHELL=/bin/bash
SUDO_GID=1000
JAVA_HOME=/opt/java
SUDO_COMMAND=/bin/bash

Command to check: env
Step10:Create the file dockerfile

FROM ubuntu: 18.04

#FROM is pre defined dockerfile key word given docker
FROM key word is used to set base image on top of it we want to place our app(soft) + dep
#FROM key word run the ubuntu image as container & execute all below on that container
RUN mkdir /opt/java
#RUN is pre defined dockerfile key word given docker
RUN key word is used to run a command / make changes on top of oslib during the image build process
COPY jdk-11.0.17 /opt/java/
COPY is pre defined dockerfile key word given docker
to copy the contents from a local dir/file to cont path
COPY src(vm/file-on-vm) dest(path-inside-oslib(baseimage))
ENV JAVA_HOME /opt/java
ENV JAVA_VER 1.11
RUN mkdir /opt/tomcat
COPY apache-tomcat-9.0.21 /opt/tomcat/
ENV TOMCAT_HOME /opt/tomcat

COPY sampleapp.war /opt/tomcat/webapps/
EXPOSE 8080
CMD ["/opt/tomcat/bin/catalina.sh", "run"]

Step2: Save the file
Step 3: Run the below command to execute the file:

docker build --file dockerfile --tag cizaar:v1 /root(where we created the files like that java,apache and sample)

```
root@ip-172-31-6-98:~# docker image ls
REPOSITORY   TAG       IMAGE ID       CREATED          SIZE
cizaar       v1        8f14eed69211   6 seconds ago    361MB
<none>       <none>    02585edb2ea0   9 minutes ago    346MB
nginx        latest    92b11f67642b   6 weeks ago      187MB
ubuntu       18.04     f9a80a55f492   10 months ago    63.2MB
root@ip-172-31-6-98:~# docker build --file dockerfile --tag cizaar:v1 /root
[+] Building 0.2s (11/11) FINISHED                                                                  docker:default
 => [internal] load build definition from dockerfile                                                         0.0s
 => => transferring dockerfile: 922B                                                                         0.0s
 => [internal] load metadata for docker.io/library/ubuntu:18.04                                              0.0s
 => [internal] load .dockerignore                                                                            0.0s
 => => transferring context: 2B                                                                              0.0s
 => [1/6] FROM docker.io/library/ubuntu:18.04                                                                0.0s
 => [internal] load build context                                                                            0.1s
 => => transferring context: 92.81kB                                                                         0.1s
 => CACHED [2/6] RUN mkdir /opt/java                                                                         0.0s
 => CACHED [3/6] COPY jdk-11.0.17 /opt/java/                                                                 0.0s
 => CACHED [4/6] RUN mkdir /opt/tomcat                                                                       0.0s
 => CACHED [5/6] COPY apache-tomcat-9.0.21 /opt/tomcat/                                                      0.0s
 => CACHED [6/6] COPY sampleapp.war /opt/tomcat/webapps/                                                     0.0s
 => exporting to image                                                                                       0.0s
 => => exporting layers                                                                                      0.0s
 => => writing image sha256:8f14eed69211464dbb0bb28abf1bb1a2e9732b2fa9f9afc62b96b8e3610f2b5c                 0.0s
 => => naming to docker.io/library/cizaar:v1                                                                 0.0s
root@ip-172-31-6-98:~# docker container run -d -P cizaar:v1
47c5d29eac4f955e78529bf4595a8e0d357ede29194e18a7f8928f279220329a
root@ip-172-31-6-98:~#
```

Step 13:NOw we can see the sampleapp image is created
docker image ls

Step14: Now we are running the container
docker container run -d -P cizaar:v1

```
root@ip-172-31-6-98:~# docker container run -d -P cizaar:v1
47c5d29eac4f955e78529bf4595a8e0d357ede29194e18a7f8928f279220329a
root@ip-172-31-6-98:~#
```

Step15: To list the running container

docker container ps

Step16: To work on the container :
docker exec -it 47c5d29eac4f /bin/bash

IN the above code we are using the port 8080 so how we get to know that we can see with the below command:

Docker image inspect nginx:latest

With the above command we got the output so with the output we can check in the expose for ex:

```
"User": "",
        "AttachStdin": false,
        "AttachStdout": false,
        "AttachStderr": false,
        "ExposedPorts": {
            "8080/tcp": {}
        },
```

==========END==========