WAY2LEARN

LAB- Docker

Step 1 -- Installing Docker toolbox and using it

1) Download docker toolbox from the below URL and install it. :
https://www.docker.com/products/docker-toolbox

2) Open Docker Quick start terminal and execute the following command :

```
docker run hello-world
```

3)docker pull command pulls an image from the docker registry and saves it in our system.

execute the following command :

docker pull ubuntu

4) Following command lists all the images downloaded :

docker images

5) To run a docker container based in an image, execute the following command :

docker run ubuntu

When you call run, the Docker client finds the image (ubuntu in this case), loads up the container and then runs a command in that container. When we run docker run ubuntu, we didn't provide a command, so the container booted up, ran an empty command and then exited.

Run the following command and observe the output :

docker run ubuntu echo "Hello from Ubuntu"

6) Following command shows all the containers running currently :

docker ps

Following shows all the containers which have exited also:

docker ps -a

Notice the status column

7) Running the run command with the -it flags attaches us to an interactive tty in the container. Now we can run as many commands in the container as we want

docker run -it ubuntu sh

Now you can execute any shell commands .

Finally execute "exit" command to come out of shell

8) Following command can be executed to remove the docker containers :

docker rm containerid

How to get Container Ids ? (execute docker ps -a)

To remove all conatiners whose status is exited , execute the following command :

docker rm $(docker ps -a -q -f status=exited)

Can you find out the meaning of -a -q -f using help?

Step 2 -- deploying static website inside apache2

1) Go to folder 01-static-webapp-in-apache2-start given to you under docker-labs

Under public_html folder, open index.html and observe it.

We want this index file to be served by apache server running inside a docker container

2) Create a file with name "Dockerfile". (Make sure that extension of file is not .txt)

Write following lines in Dockerfile

FROM httpd:2.4
COPY ./public_html/ /usr/local/apache2/htdocs/

3) Create the image using below command :

docker build -t siva-apache2 .

In the above command, siva-apache2 is the image name. Feel free to use any name.

4) Create a docker container running this image using the below command :

docker run siva-apache2

Above command will run an instance of siva-apache2 image. By default , instance name is same as image name

If we want to give our own container name we can use --name option as shown below :
docker run --name apache2instance siva-apache2

Also, this command is running in foreground. If we want to run the container as daemon, we have to use -d option as shown below :

docker run -d --name apache2instance siva-apache2

In this case, the client is not exposing any ports so we need to re-run the docker run command to publish ports

docker run -d -P --name apache2instance siva-apache2

After running the above command, run the below command to see the ports exposed

docker port apache2instance

Above command will display the ports like below :

$ docker port apache2instance
80/tcp -> 0.0.0.0:32772

Now try to give request to localhost:32772 on browser. You should not see the result if you are running docker on windows . You need to get the ip assigned to the docker machine using below command

docker-machine ip default

Now use the ip given and give request using browser. You should see the result

But now we dont want a random port. We want to assign a port when we are running the docker container. So, use the below command :

docker run -d -p 6060:80 --name apache2instance siva-apache2

Note : Stop the previously running instance using docker stop containerid. Otherwise you may get an error

Now you can use 6060 instead of random port.

To stop the docker container, use the below command :

docker stop containerid

Step 3 -- Running a jar in docker container

1) cd to 02-running-jar-start inside docker-labs folder and observe that there is eureka.jar.

Open cmd and execute java -jar eureka.jar . Observe that it starts an embedded tomcat at 4001.

open browser and give a request to http://localhost:4001. You should observe eureka server console.

Now we want to start the same inside docker.

Firstly, stop the java command in cmd.

2) Create a file with name Dockerfile inside 02-running-jar-start folder

Write the following inside the file and save it.

```
FROM anapsix/alpine-java
COPY eureka.jar /home/eureka.jar
CMD ["java","-jar","/home/eureka.jar"]
```

3) Build the image using below command :

docker build -t eureka

4) Now create a docker container instance for image using below command :

docker run -d -p 4001:4001 --name eurekainstance1 eureka

5) To restart the container , you can use the below command:

docker restart eurekainstance1

6) If your program has changed and you want to rebuild your image, you the below commands :

docker rmi eureka
docker build -t eureka

Step 4 -- Running a war in docker container

1) cd to **03-deploying-war-intomcat-start** inside docker-labs folder and observe that there is helloapp.war

Deploy helloapp.war inside local tomcat

Now give a request to http://localhost:8080/helloapp and observe that it displays "Hello World from WebApp inside Docker"

Now, stop the Tomcat.
We want to deploy the same application inside tomcat running in docker container.

2) Create a file with name Dockerfile inside **03-deploying-war-intomcat-start** folder

Write the following inside the file and save it.

From tomcat:8-jre8
ADD helloapp.war /usr/local/tomcat/webapps/

3) Build the image using below command :

docker build -t helloapp .

4) Now create a docker container instance for image using below command :

docker run -d -p 8080:8080 --name helloappinstance1 helloapp

5) To stop the docker container instance, run the following command :

docker stop helloappinstance1

WAY2LEARN

Step 4 -- Running a war dynamically in docker container

1) Create a file with name Dockerfile inside **04-deploying-dynamic-war-intomcat-start** folder

Write the following inside the file and save it.

From tomcat:8-jre8

2) Build the image using below command :

docker build -t helloapp .

3) Now create a docker container instance for image using below command :

docker run -it -p 8080:8080 -v /C/microservices-latest/docker-labs/04-deploying-dynamic-war-intomcat-start:/usr/local/tomcat/webapps --name helloappinstance2 helloapp

Known Issue in Windows : Volume Mounting is not working in Windows as of now. It might be fixed in later versions!!!!!!!!!!

Step 5 -- Configuring DOCKER specific environment variables so that we can execute docker commands from cmd of windows

If we want to run the docker commands from Windows Command prompt directly instead of docker toolbox , following environment variables have to be set :

Firstly, go to Docker tool box window and execute following command : docker-machine env

This will display all the environment variables .

Configure same environment variables in windows.

Now you can execute any docker commands directly from command prompt.

Try them once.

From STS if you want to do a maven build , Right click on Project , RUN AS --> Maven build ... Click on environment tab and add the same environment variables

Step 5 -- Creating a docker image using Maven

If you want to build a docker image using maven , u have to configure following plugin in pom.xml

```
<plugin>
                        <groupId>com.spotify</groupId>
                        <artifactId>docker-maven-plugin</artifactId>
                        <version>0.4.11</version>
                        <configuration>
                              <imageName>way2learn/${project.artifactId}</imageName>

      <dockerDirectory>${project.basedir}/src/main/docker</dockerDirectory>
                              <resources>
                                    <resource>

      <directory>${project.build.directory}</directory>

      <include>${project.build.finalName}.jar</include>
                                    </resource>
                              </resources>
                        </configuration>
                  </plugin>
```

You have to keep your docker file in `${project.basedir}/src/main/docker`