Lab 8: Creating a Docker Image

In this lab, i'll explain how can we run our QA automation projects in docker containers. I'll take an simple example of a Java, Junit and Maven based test project.

Before running the project in docker container we will see how to run it on local.

Prerequisites:

1. Java, maven should be installed and set in environment variables. (To run it locally)
2. Docker should be installed on your machine. (To run it in docker container)
3. Basic knowledge about docker (how to build images and run containers)

Instructions to run on local:

1. Go to following directory in your lab machine:

```
C:\Users\fenago\Desktop\advanced-selenium-java\Lab08
```

2. open a terminal window and cd to the project directory.
3. run **mvn clean test** to start the test execution.

Once the command is run, execution will start and maven will start downloading the dependencies and finally you will see following text printed on the terminal.

```
This test will run in docker container
```

Now that we have run the project in local machine, we will see how to run in docker container.

Instructions to run in docker container:

1. Build a docker image using the Dockerfile.
2. Run a container using the image built in the first step.
3. Run **mvn clean test** command inside the container. (This command can be run using the docker file itself. We will see it in coming labs)

First lets see what is a docker file and what it is used for.

Dockerfile explanation:

if you have noticed already there is a Dockerfile in the root directory of the project which contains the instructions to build the image. I have taken a base image (maven:3.6-jdk-8-slim) which has both java and maven in it.

Please go through the explanation of each and every instructions written in the docker file below to know what they do.

```
FROM maven:3.6-jdk-8-slim
WORKDIR /sample
COPY src /sample/src
```

```
COPY pom.xml /sample
```

1. **FROM** command will take 'maven:3.6-jdk-8-slim' as a base image.
2. **WORKDIR** /sample will create a directory **/sample**inside the docker container which will act as a project directory.
3. **COPY**src /sample/src****will copy the **src** folder from your local machine to **/sample/src** directory inside the docker container.

4. **COPY** pom.xml /sample will copy the **pom.xml**file from your local machine to **/sample**directory inside the docker container.

Now we will see how to build a image using the docker file.

1. open a terminal window or cmd.
2. run the following command to build the image from docker file.

```
docker build -t samplemaven:latest .
```

build command is to build the image

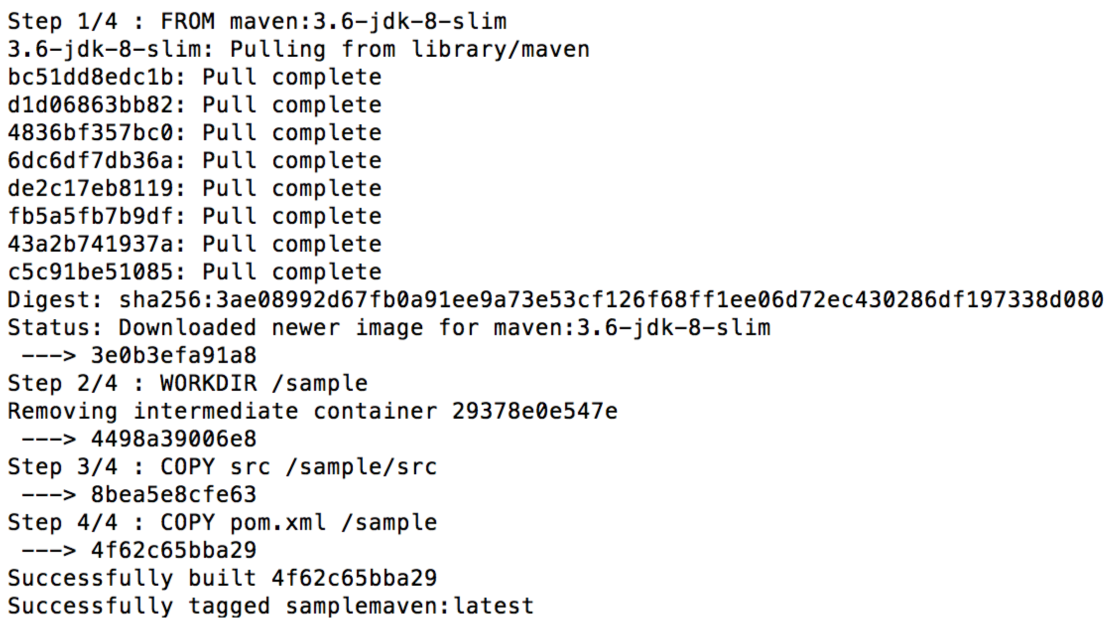
-t is used to tag the image

samplemavenis the name of the image that we have given

latest is the tag that we have given to the image

. represents that our docker file is kept at the root location of the project direcctory

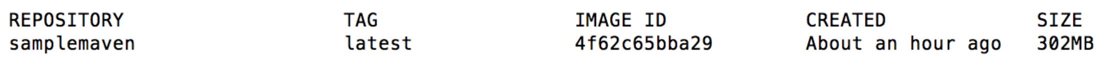
once the command is run you will see following output on the terminal



```
Step 1/4 : FROM maven:3.6-jdk-8-slim
3.6-jdk-8-slim: Pulling from library/maven
bc51dd8edc1b: Pull complete
d1d06863bb82: Pull complete
4836bf357bc0: Pull complete
6dc6df7db36a: Pull complete
de2c17eb8119: Pull complete
fb5a5fb7b9df: Pull complete
43a2b741937a: Pull complete
c5c91be51085: Pull complete
Digest: sha256:3ae08992d67fb0a91ee9a73e53cf126f68ff1ee06d72ec430286df197338d080
Status: Downloaded newer image for maven:3.6-jdk-8-slim
 ---> 3e0b3efa91a8
Step 2/4 : WORKDIR /sample
Removing intermediate container 29378e0e547e
 ---> 4498a39006e8
Step 3/4 : COPY src /sample/src
 ---> 8bea5e8cfe63
Step 4/4 : COPY pom.xml /sample
 ---> 4f62c65bba29
Successfully built 4f62c65bba29
Successfully tagged samplemaven:latest
```

once all steps are completed, a image will be build with the name **samplemaven** in your local.

Now run **docker images**to see the images on your local. You should see the following entry along with other images (if you have any already)



```
REPOSITORY                    TAG                 IMAGE ID            CREATED             SIZE
samplemaven                   latest              4f62c65bba29        About an hour ago   302MB
```

Now that the image is built and saved on your local. We will run the container using this image by running the following command.

```
docker run -it --name samplecontainer samplemaven:latest /bin/bash
```

here **-i**t represents that we are running the container in the **interactive mode**

--name we are using to name the container as**samplecontainer**

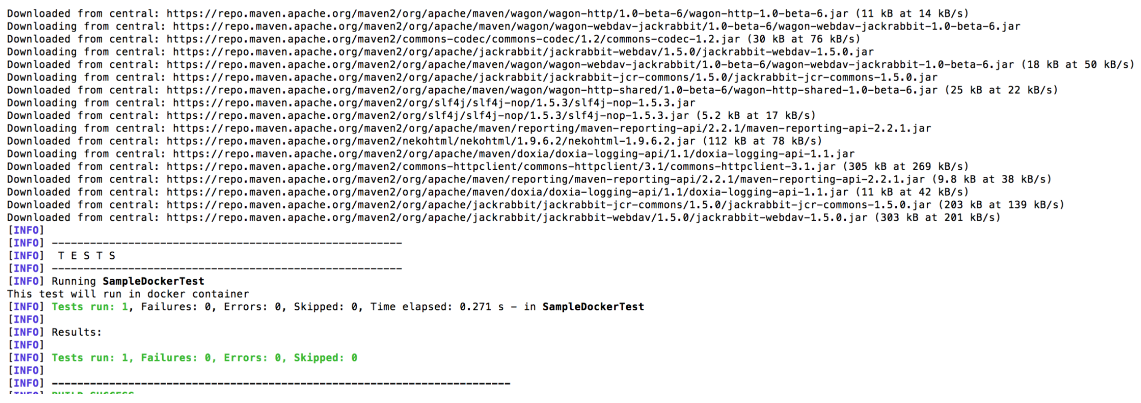
samplemaven:latestis the name of the image that we created in step 1 which we want to run

/bin/bash is the argument that we are passing to open the bash terminal once the container is started

After running this command container will be started and you will be inside the /sample directory by default.

If you run ls command here you will see src and pom.xml files which we copied using the docker file.

Now run **mvn clean test** command to start the test execution. It will start downloading the dependencies from maven and finally will execute the test cases.



```
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/wagon/wagon-http/1.0-beta-6/wagon-http-1.0-beta-6.jar (11 kB at 14 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/wagon/wagon-webdav-jackrabbit/1.0-beta-6/wagon-webdav-jackrabbit-1.0-beta-6.jar
Downloaded from central: https://repo.maven.apache.org/maven2/commons-codec/commons-codec/1.2/commons-codec-1.2.jar (30 kB at 76 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/jackrabbit/jackrabbit-webdav/1.5.0/jackrabbit-webdav-1.5.0.jar
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/wagon/wagon-webdav-jackrabbit/1.0-beta-6/wagon-webdav-jackrabbit-1.0-beta-6.jar (18 kB at 50 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/jackrabbit/jackrabbit-jcr-commons/1.5.0/jackrabbit-jcr-commons-1.5.0.jar
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/wagon/wagon-http-shared/1.0-beta-6/wagon-http-shared-1.0-beta-6.jar (25 kB at 22 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/slf4j/slf4j-nop/1.5.3/slf4j-nop-1.5.3.jar
Downloaded from central: https://repo.maven.apache.org/maven2/org/slf4j/slf4j-nop/1.5.3/slf4j-nop-1.5.3.jar (5.2 kB at 17 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/reporting/maven-reporting-api/2.2.1/maven-reporting-api-2.2.1.jar
Downloaded from central: https://repo.maven.apache.org/maven2/nekohtml/nekohtml/1.9.6.2/nekohtml-1.9.6.2.jar (112 kB at 78 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/doxia/doxia-logging-api/1.1/doxia-logging-api-1.1.jar
Downloaded from central: https://repo.maven.apache.org/maven2/commons-httpclient/commons-httpclient/3.1/commons-httpclient-3.1.jar (305 kB at 269 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/reporting/maven-reporting-api/2.2.1/maven-reporting-api-2.2.1.jar (9.8 kB at 38 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/doxia/doxia-logging-api/1.1/doxia-logging-api-1.1.jar (11 kB at 42 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/jackrabbit/jackrabbit-jcr-commons/1.5.0/jackrabbit-jcr-commons-1.5.0.jar (203 kB at 139 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/jackrabbit/jackrabbit-webdav/1.5.0/jackrabbit-webdav-1.5.0.jar (303 kB at 201 kB/s)
[INFO]
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running SampleDockerTest
This test will run in docker container
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.271 s - in SampleDockerTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
```

This way we can run our test cases in docker container.