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

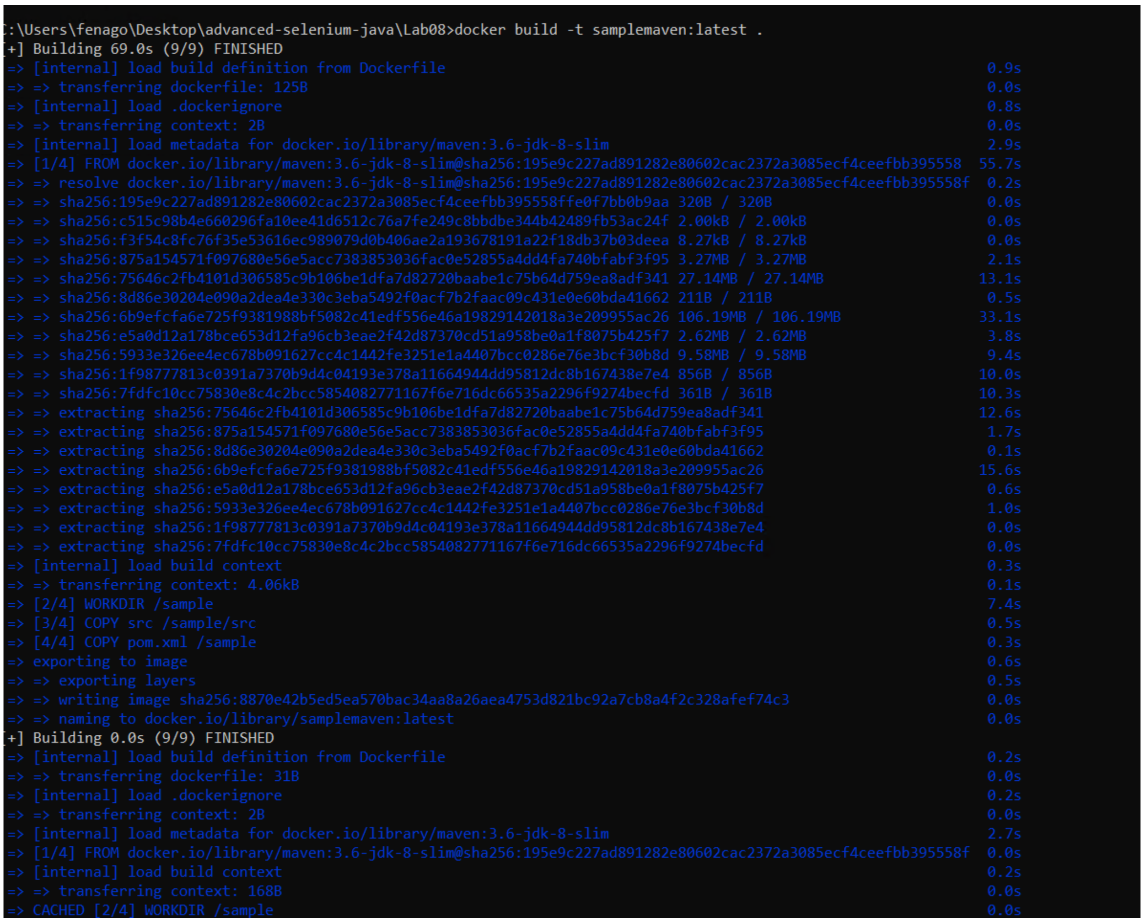
-t is used to tag the image

samplemaven is 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



```
:\Users\fenago\Desktop\advanced-selenium-java\Lab08>docker build -t samplemaven:latest .
[+] Building 69.0s (9/9) FINISHED
=> [internal] load build definition from Dockerfile                                                      0.9s
=> => transferring dockerfile: 125B                                                                      0.0s
=> [internal] load .dockerignore                                                                         0.8s
=> => transferring context: 2B                                                                           0.0s
=> [internal] load metadata for docker.io/library/maven:3.6-jdk-8-slim                                   2.9s
=> [1/4] FROM docker.io/library/maven:3.6-jdk-8-slim@sha256:195e9c227ad891282e80602cac2372a3085ecf4ceefbb395558  55.7s
=> => resolve docker.io/library/maven:3.6-jdk-8-slim@sha256:195e9c227ad891282e80602cac2372a3085ecf4ceefbb395558f  0.2s
=> => sha256:195e9c227ad891282e80602cac2372a3085ecf4ceefbb395558ffe0f7bb0b9aa 320B / 320B                 0.0s
=> => sha256:c515c98b4e660296fa10ee41d6512c76a7fe249c8bbdbe344b42489fb53ac24f 2.00kB / 2.00kB             0.0s
=> => sha256:f3f54c8fc76f35e53616ec989079d0b406ae2a193678191a22f18db37b03deea 8.27kB / 8.27kB             0.0s
=> => sha256:875a154571f097680e56e5acc7383853036fac0e52855a4dd4fa740bfabf3f95 3.27MB / 3.27MB             2.1s
=> => sha256:75646c2fb4101d306585c9b106be1dfa7d82720baabe1c75b64d759ea8adf341 27.14MB / 27.14MB          13.1s
=> => sha256:8d86e30204e090a2dea4e330c3eba5492f0acf7b2faac09c431e0e60bda41662 211B / 211B                 0.5s
=> => sha256:6b9efcfa6e725f9381988bf5082c41edf556e46a19829142018a3e209955ac26 106.19MB / 106.19MB        33.1s
=> => sha256:e5a0d12a178bce653d12fa96cb3eae2f42d87370cd51a958be0a1f8075b425f7 2.62MB / 2.62MB             3.8s
=> => sha256:5933e326ee4ec678b091627cc4c1442fe3251e1a4407bcc0286e76e3bcf30b8d 9.58MB / 9.58MB             9.4s
=> => sha256:1f98777813c0391a7370b9d4c04193e378a11664944dd95812dc8b167438e7e4 856B / 856B                10.0s
=> => sha256:7fdfc10cc75830e8c4c2bcc5854082771167f6e716dc66535a2296f9274becfd 361B / 361B                10.3s
=> => extracting sha256:75646c2fb4101d306585c9b106be1dfa7d82720baabe1c75b64d759ea8adf341                 12.6s
=> => extracting sha256:875a154571f097680e56e5acc7383853036fac0e52855a4dd4fa740bfabf3f95                  1.7s
=> => extracting sha256:8d86e30204e090a2dea4e330c3eba5492f0acf7b2faac09c431e0e60bda41662                  0.1s
=> => extracting sha256:6b9efcfa6e725f9381988bf5082c41edf556e46a19829142018a3e209955ac26                 15.6s
=> => extracting sha256:e5a0d12a178bce653d12fa96cb3eae2f42d87370cd51a958be0a1f8075b425f7                  0.6s
=> => extracting sha256:5933e326ee4ec678b091627cc4c1442fe3251e1a4407bcc0286e76e3bcf30b8d                  1.0s
=> => extracting sha256:1f98777813c0391a7370b9d4c04193e378a11664944dd95812dc8b167438e7e4                  0.0s
=> => extracting sha256:7fdfc10cc75830e8c4c2bcc5854082771167f6e716dc66535a2296f9274becfd                  0.0s
=> [internal] load build context                                                                         0.3s
=> => transferring context: 4.06kB                                                                       0.1s
=> [2/4] WORKDIR /sample                                                                                 7.4s
=> [3/4] COPY src /sample/src                                                                            0.5s
=> [4/4] COPY pom.xml /sample                                                                            0.3s
=> exporting to image                                                                                    0.6s
=> => exporting layers                                                                                   0.5s
=> => writing image sha256:8870e42b5ed5ea570bac34aa8a26aea4753d821bc92a7cb8a4f2c328afef74c3              0.0s
=> => naming to docker.io/library/samplemaven:latest                                                     0.0s
[+] Building 0.0s (9/9) FINISHED
=> [internal] load build definition from Dockerfile                                                      0.2s
=> => transferring dockerfile: 31B                                                                       0.0s
=> [internal] load .dockerignore                                                                         0.2s
=> => transferring context: 2B                                                                           0.0s
=> [internal] load metadata for docker.io/library/maven:3.6-jdk-8-slim                                   2.7s
=> [1/4] FROM docker.io/library/maven:3.6-jdk-8-slim@sha256:195e9c227ad891282e80602cac2372a3085ecf4ceefbb395558f  0.0s
=> [internal] load build context                                                                         0.2s
=> => transferring context: 168B                                                                         0.0s
=> CACHED [2/4] WORKDIR /sample                                                                          0.0s
```

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 **-it** represents that we are running the container in the **interactive mode**

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

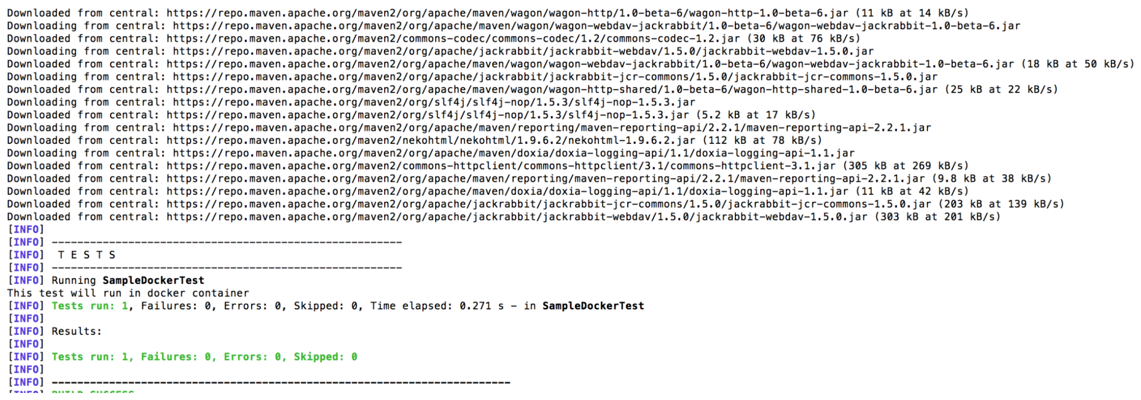
samplemaven:latest is the name of the image that we created in step 1 which we want to run

/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.