01: Docker tutorial with Java & Maven

Posted on May 17, 2018

Pre-requisite: Docker is installed on your machine for Mac OS X (E.g. $ brew cask install docker) or Windows 10. Docker interview Q&As.

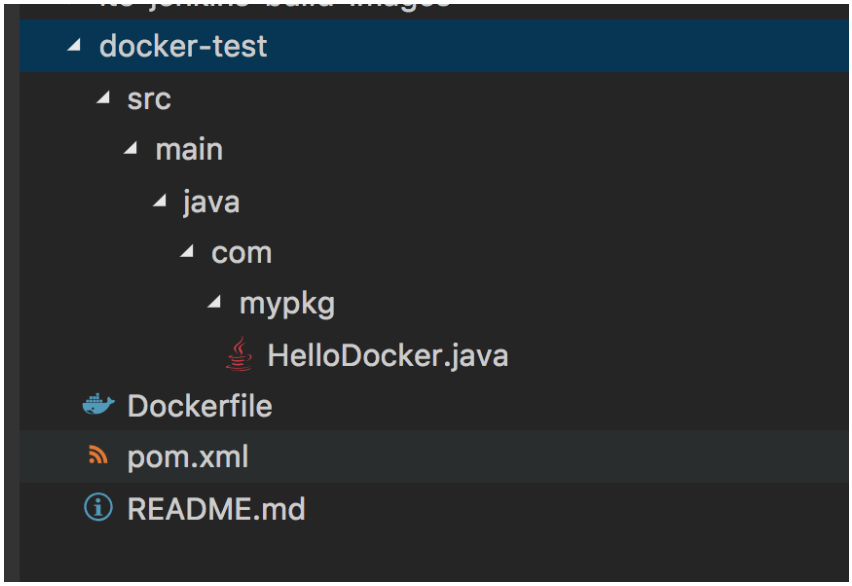
Step 1: Create a Java project "**docker-test**" with "**HelloDocker.java**" file under "src/main/java" in a package named "com/mypkg", and "**pom.xml**" in the base project folder. Using the Visual Studio Code editor.

300+ Java Interview FAQs

- 300+ Java FAQs 🔥
- 16+ Java Key Areas Q&As
- 150+ Java Architect FAQs
- 80+ Java Code Quality Q&As
- 150+ Java Coding Q&As

300+ Big Data Interview FAQs

- 300+ Big Data FAQs 🔥
- Tutorials - Big Data
  - TUT - ⏯ Starting Big Data
  - TUT - Starting Spark & Scala



Basic Mavenized Java folder structure with Dockerfile

HelloDocker.java

```
1  package com.mypkg;
2
3  public class HelloDocker {
4    public static void main(String[] args) throws Ex
5      System.out.println("Hello Docker");
6    }
7  }
8
```

pom.xml

```
1
2  <project xmlns="http://maven.apache.org/POM/4.0.0
3  xsi:schemaLocation="http://maven.apache.org/POM/4
4    <modelVersion>4.0.0</modelVersion>
5    <groupId>com.myproject</groupId>
6    <artifactId>hellodocker</artifactId>
7    <version>1.0</version>
8
9    <build>
10     <plugins>
11       <plugin>
12         <!-- Build an executable JAR -->
13         <groupId>org.apache.maven.plugins</groupId
14         <artifactId>maven-jar-plugin</artifactId>
15         <version>3.1.0</version>
16         <configuration>
17           <archive>
18             <manifest>
```

TUT - Starting with Python
TUT - Kafka
TUT - Pig
TUT - Apache Storm
TUT - Spark Scala on Zeppelin
TUT - Cloudera
TUT - Cloudera on Docker
TUT - File Formats
TUT - Spark on Docker
TUT - Flume
TUT - Hadoop (HDFS)
TUT - HBase (NoSQL)
TUT - Hive (SQL)
TUT - Hadoop & Spark
TUT - MapReduce
TUT - Spark and Scala
TUT - Spark & Java
TUT - PySpark on Databricks
TUT - Zookeeper

```
            <addClasspath>true</addClasspath>
            <classpathPrefix>lib/</classpathPre
            <mainClass>com.mypkg.HelloDocker</m
          </manifest>
        </archive>
      </configuration>
    </plugin>
  </plugins>
 </build>
</project>
```

Step 2: Create a "**Dockerfile**" in the project base directory.

Dockerfile

```

FROM maven:3.5-jdk-8-alpine
MAINTAINER java-success.com

WORKDIR /app

CMD ["/bin/sh"]
```

Step 3: Docker has two steps. Firstly, "**build**" an image based on the "Dockerfile" and then "**run**" the container, which is an instance of the image.

From the docker-test folder build the image

it uses the "Dockerfile" in the "." (i.e. current directory)

```

$ docker build --tag=docker-test .
```

once the docker image is built, you can list it with

```

$ docker images
```

```

REPOSITORY                    TAG
```

```
3 docker-test                    latest
```

It has built an image of size 119MB. The image name will be “docker-test:latest”

You can remove an image with “docker rmi 25a4f7356b10”.

From the docker-test create a container, which is an instance of the image

```
1
2 $ docker run --rm -it -v "$(pwd):/app:consistent" (
```

Present working directory is mapped to Docker container’s “/app” and synchronised.

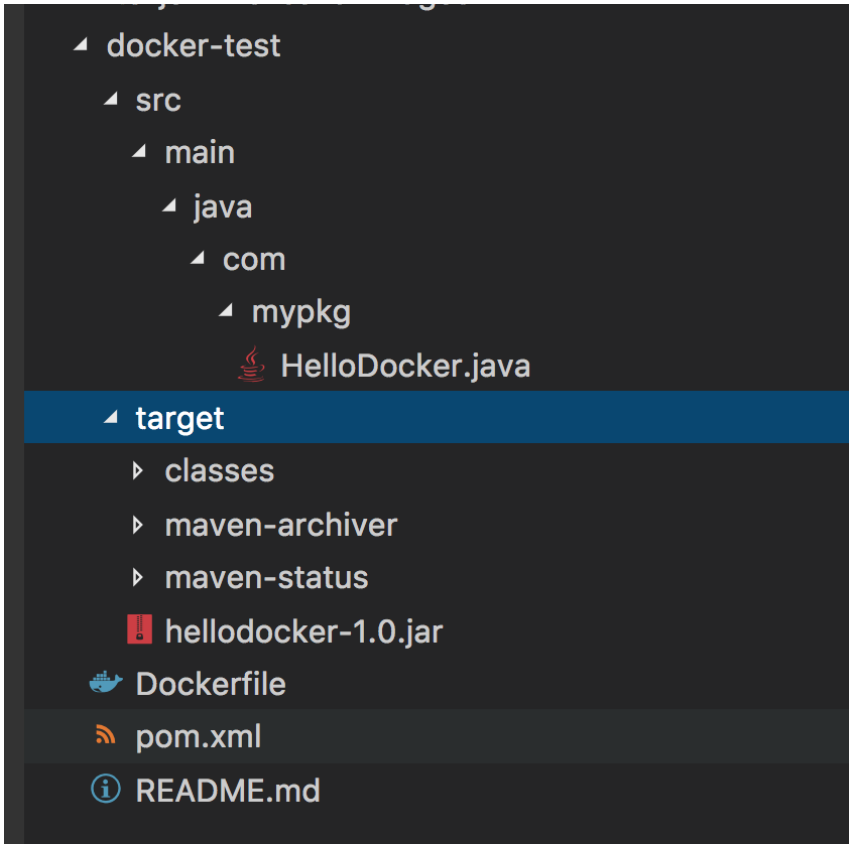
Now, you are in the Docker container prompt

```
1
2 /app #
```

Step 4: You can run “mvn package” to build the jar file “hellodocker-1.0.jar” in “target” folder.

```
1
2 /app # mvn -Dmaven.repo.local="/app/mvn-repository"
```

After packaging, there will be a target folder with “hellodocker-1.0.jar” file.



After packaging.

"-v "$(pwd):/app:consistent" keeps the folders & files in the Docker container in the "/app" folder in sync with the current working directory "docker-test" shown above.

Step 5: Run the "hellodocker-1.0.jar"

```
/app # java -jar target/hellodocker-1.0.jar
```

Output:

```
Hello Docker
```

You could also have directly run

Create an image

```
1
2 docker run --rm -it -v "$(pwd):/app:consistent" do
```

Create a container, which is an instance of the image

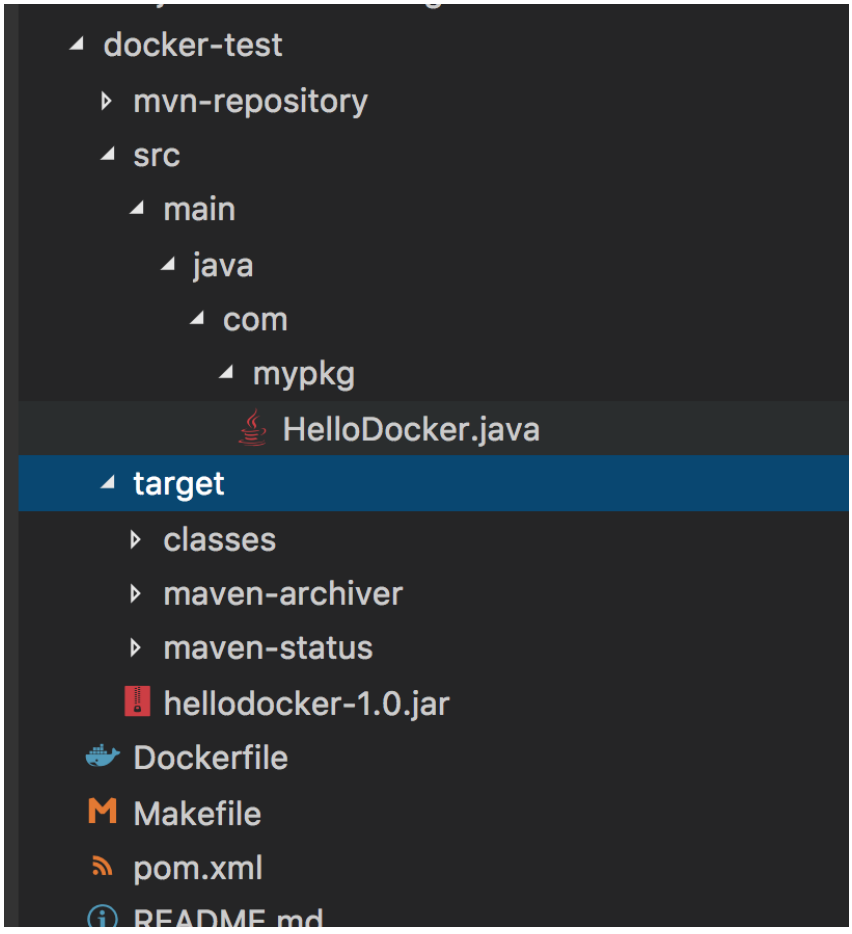
```
1
2 docker run --rm -it -v "$(pwd):/app:consistent" do
```

Notes: Look at the “Docker hub” for the images that you can use.

GNU make file to automate docker runs

Assumes that GNU Make is installed on your machine.

Step 6: Create a “Makefile” in the base project folder.



Docker with makefile

Makefile

```
help:
    @cat $(MAKEFILE_LIST) | grep -e "^[a-zA-Z_\-]*

docker-build:
    docker build --tag=docker-test .

docker-run: docker-build
    docker run --rm -it -v "$(pwd):/app:consisten

app-run: docker-build
    docker run --rm -it -v "$(shell pwd):/app:cons
    mvn -Dmaven.repo.local="/app/mvn-repository" p
    java -jar target/hellodocker-1.0.jar
```

You can run the commands like:

```
$ make
$ make docker-run
$ make app-run
```

If you want to run the app, just execute

```
$ make app-run
```

‹ 17: Spark interview Q&As with coding examples in pyspark (i.e. python)

02: Apache Spark – local mode on Docker tutorial with Java & Maven ›

Disclaimer