20: Docker Tutorial: Apache Spark (spark-submit) in Java on Cloudera quickstart

Posted on June 7, 2019

This extends Docker Tutorial: BigData on Cloudera quickstart via Docker.

Step 1: Run the container on a command line.

```
~/projects/docker-hadoop]$ docker run --hostname=qu
--privileged=true -t -i -v /Users/arulkumarankumar
--publish-all=true -p 8888:8888 -p 80:80 -p 7180:7
```

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

Step 2: Java is already installed as part of the "cloudera/quickstart" docker image.

```
[root@quickstart /]# java -version
java version "1.7.0_67"
Java(TM) SE Runtime Environment (build 1.7.0_67-b01
Java HotSpot(TM) 64-Bit Server VM (build 24.65-b04
```

update-alternatives

If you have multiple versions or installations of Java, you can list them as shown below:

```
[root@quickstart /]# update-alternatives --config j
```

Install maven

Step 3: Download maven from "http://mirror.ventraip.net.au/apache/maven/maven-3/3.6.1/binaries/apache-maven-3.6.1-bin.tar.gz"

```
[root@quickstart /]# sudo curl -L -O http://mirror
```

```
[root@quickstart /]# cp apache-maven-3.6.1-bin.tar
[root@quickstart /]# cd /opt
[root@quickstart /]# sudo tar xvfz apache-maven-3.(
[root@quickstart /]# sudo rm -f apache-maven-3.6.1-
```

Step 4: Add it to your profile so that you can run the "mvn" command. Add it to "**.bashrc**" if you are a root user or to "**.bash_profile**" if you are a any other user.

```
[root@quickstart opt]# vi ~/.bash_profile
```

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

800+ Java Interview Q&As

300+ Core Java Q&As
300+ Enterprise Java Q&As
150+ Java Frameworks Q&As
120+ Companion Tech Q&As
Tutorials - Enterprise Java

add the following lines at the end of "~/.bash_profile"

```
M3_HOME=/opt/apache-maven-3.6.1
export MAVEN_OPTS=-Dhttps.protocols=TLSv1,TLSv1.1,
export PATH=$PATH:$M3_HOME/bin

```

Apply the changes with the "**source**" command.

```
[root@quickstart opt]# source  ~/.bash_profile
```

```
[root@quickstart opt]# mvn -version
Apache Maven 3.6.1 (d66c9c0b3152b2e69ee9bac180bb8f
Maven home: /opt/apache-maven-3.6.1
Java version: 1.7.0_67, vendor: Oracle Corporation
Default locale: en_US, platform encoding: UTF-8
OS name: "linux", version: "4.9.125-linuxkit", arc
[root@quickstart opt]#

```

Now Maven is installed.

Note: Even though the maven is already in the docker image we have installed the latest version to a different location "/opt/".

```
[root@quickstart /]# ls -ltr /usr/local/apache-mav
total 4
drwxr-xr-x 6 root root 4096 Apr  6  2016 apache-ma
[root@quickstart /]#
```

Create a simple maven project structure

Step 5: Create "**projects**" folder and create a maven project structure.

```
[root@quickstart ~]# cd /
```

```
[root@quickstart ~]# mkdir /projects
[root@quickstart ~]# cd /projects

```

Create a maven project structure.

```
[root@quickstart projects]# mvn archetype:generate
   -DartifactId=my-app \
   -DarchetypeArtifactId=maven-archetype-quickstar
   -DinteractiveMode=false

```

Give it a few minutes.

```
[root@quickstart projects]# tree my-app
my-app
├── pom.xml
└── src
    ├── main
    │   └── java
    │       └── com
    │           └── mycompany
    │               └── app
    │                   └── App.java
    └── test
        └── java
            └── com
                └── mycompany
                    └── app
                        └── AppTest.java

11 directories, 3 files
[root@quickstart projects]#

```

Create a simple Spark job in Java

Step 6: Add the Spark library to maven pom.xml file.

```
[root@quickstart projects]# cd my-app
[root@quickstart my-app]# vi pom.xml

```

The pom.xml file should look like

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xsi:schemaLocation="http://maven.apache.org/POM/
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.mycompany.app</groupId>
  <artifactId>my-app</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>my-app</name>
  <url>http://maven.apache.org</url>
  <dependencies>

    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
    <!-- https://mvnrepository.com/artifact/org.ap
    <dependency>
        <groupId>org.apache.spark</groupId>
        <artifactId>spark-core_2.10</artifactId>
        <version>1.6.0</version>
    </dependency>

  </dependencies>
</project>

```

Step 7: Create a simple Spark job in Java.

```
[root@quickstart my-app]# vi src/main/java/com/myc
```

The “SimpleSpark.java” file looks like:

```
package com.mycompany.app;

import org.apache.spark.api.java.*;
import org.apache.spark.SparkConf;
import java.util.*;
import org.apache.spark.api.java.function.Functio

public class SimpleSpark {

    public static void main (String[] args) {
        SparkConf conf = new SparkConf().setAppNa
        JavaSparkContext sc = new JavaSparkContex

```

```
14          List<String> data = Arrays.asList("John",
15          JavaRDD<String> rdd = sc.parallelize(data
16          System.out.println(rdd.collect());
17
18      }
19
20  }
21
```

compile & package it as a jar file with maven

Step 8: Compile & package it as a jar file.

```
1  [root@quickstart my-app]# mvn clean package
```

```
1  [root@quickstart my-app]# tree
2  .
3  ├── pom.xml
4  ├── src
5  │   ├── main
6  │   │   └── java
7  │   │       └── com
8  │   │           └── mycompany
9  │   │               └── app
10 │   │                   ├── App.java
11 │   │                   └── SimpleSpark.java
12 │   └── test
13 │       └── java
14 │           └── com
15 │               └── mycompany
16 │                   └── app
17 │                       └── AppTest.java
18 └── target
19     ├── classes
20     │   └── com
21     │       └── mycompany
22     │           └── app
23     │               ├── App.class
24     │               └── SimpleSpark.class
25     ├── maven-archiver
26     │   └── pom.properties
27     ├── maven-status
28     │   └── maven-compiler-plugin
29     │       ├── compile
30     │       │   └── default-compile
31     │       │       ├── createdFiles.lst
32     │       │       └── inputFiles.lst
33     │       └── testCompile
34     │           └── default-testCompile
```

```
35      |                   ├── createdFiles.lst
36      |                   └── inputFiles.lst
37      ├── my-app-1.0-SNAPSHOT.jar
38      ├── surefire-reports
39      |   ├── com.mycompany.app.AppTest.txt
40      |   └── TEST-com.mycompany.app.AppTest.xml
41      └── test-classes
42          └── com
43              └── mycompany
44                  └── app
45                      └── AppTest.class
46
47  28 directories, 15 files
48  [root@quickstart my-app]#
```

"my-app-1.0-SNAPSHOT.jar" is the packaged jar file in the "target" folder.

spark-submit to run the spark job

Step 9: Run the Spark job in the jar file via Spark-submit command.

Local client mode

```
1  [root@quickstart my-app]# spark-submit   \
2  --class com.mycompany.app.SimpleSpark  \
3  --master local \
4  --deploy-mode client \
5  target/my-app-1.0-SNAPSHOT.jar
6
```

```
1  ...
2  [John, Peter, Samuel]
3  ....
4
```

Cluster yarn mode

```
1  [root@quickstart my-app]# spark-submit   \
2  --class com.mycompany.app.SimpleSpark  \
3  --master yarn \
4  --deploy-mode cluster \
5  target/my-app-1.0-SNAPSHOT.jar
6
```

The job will be submitted to yarn and executed. In order to look at the logs you need to check the logs for the printed values.

‹ 19: Docker Tutorial: Apache Spark SQL – on Cloudera quickstart

21: Docker Tutorial: Apache Spark (spark-submit) in Scala on Cloudera quickstart ›

Disclaimer