28: Docker Tutorial: Apache Spark streaming with Kafka in Java on Cloudera quickstart

Posted on June 23, 2019

This extends 27: Docker Tutorial: Apache Kafka with Java API on Cloudera quickstart

Install Java 8

Step 1: Install Java 8.

```
[kafka@quickstart my-app]$ sudo yum install java-1
```

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

```
[kafka@quickstart my-app]$ java -version
openjdk version "1.8.0_212"
OpenJDK Runtime Environment (build 1.8.0_212-b04)
OpenJDK 64-Bit Server VM (build 25.212-b04, mixed r

```

```
[kafka@quickstart my-app]$ update-alternatives --c

There is 1 program that provides 'java'.

  Selection    Command
-----------------------------------------------
*+ 1           /usr/lib/jvm/jre-1.8.0-openjdk.x86_6


```

JAVA_HOME

Step 2: Set JAVA_HOME to Java 8 so that Maven uses Java 8.

```
[kafka@quickstart my-app]$ vi ~/.bash_profile

```

```
# .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
        . ~/.bashrc
fi

# User specific environment and startup programs

PATH=$PATH:$HOME/bin

export PATH

export JAVA_HOME=/usr/lib/jvm/java-1.8.0

PATH=$PATH:/usr/local/apache-maven/apache-maven-3

```

Activate:

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

"!$" means use the last argument which is "~/.bash_profile"

```
[kafka@quickstart my-app]$ source !$
source ~/.bash_profile

```

Check that maven is using Java 8:

```
[kafka@quickstart my-app]$ mvn -version
Apache Maven 3.0.4 (r1232337; 2012-01-17 08:44:56+(
Maven home: /usr/local/apache-maven/apache-maven-3
Java version: 1.8.0_212, vendor: Oracle Corporatio
Java home: /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.2
Default locale: en_US, platform encoding: UTF-8
OS name: "linux", version: "4.9.125-linuxkit", arc

```

pom.xml

Step 3: Add spark-core & spark-streaming dependencies, and the plugin to build **uber** jar to the pom.xml file. Also, Java 1.8 is used in Maven.

```
<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>

  <properties>
        <maven.compiler.target>1.8</maven.compile
        <maven.compiler.source>1.8</maven.compile
  </properties>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
```

```
21          <scope>test</scope>
22        </dependency>
23
24        <dependency>
25            <groupId>org.apache.kafka</groupId>
26            <artifactId>kafka-clients</artifactId>
27            <version>0.11.0.0</version>
28        </dependency>
29
30        <!-- https://mvnrepository.com/artifact/org.ap
31        <dependency>
32            <groupId>org.apache.spark</groupId>
33            <artifactId>spark-core_2.10</artifactId>
34            <version>1.6.0</version>
35        </dependency>
36
37        <dependency>
38            <groupId>org.apache.spark</groupId>
39            <artifactId>spark-streaming_2.10</artifact
40            <version>1.6.0</version>
41        </dependency>
42
43        <dependency>
44            <groupId>org.apache.spark</groupId>
45            <artifactId>spark-streaming-kafka_2.10</ar
46            <version>1.6.0</version>
47        </dependency>
48
49    </dependencies>
50
51    <!-- Build uber jar -->
52    <build>
53        <plugins>
54            <plugin>
55                <groupId>org.apache.maven.plugins</gro
56                <artifactId>maven-compiler-plugin</art
57                <version>3.6.1</version>
58                <configuration>
59                    <source>1.8</source>
60                    <target>1.8</target>
61                </configuration>
62            </plugin>
63
64            <plugin>
65               <groupId>org.apache.maven.plugins</grou
66               <artifactId>maven-shade-plugin</artifac
67               <executions>
68                  <execution>
69                     <phase>package</phase>
70                     <goals>
71                        <goal>shade</goal>
72                     </goals>
73                     <configuration>
74                        <filters>
75                           <filter>
```

```
                            <artifact>*:*</artifa
                            <excludes>
                                <exclude>META-INF/
                                <exclude>META-INF/
                                <exclude>META-INF/
                            </excludes>
                        </filter>
                    </filters>
                    <!-- Additional configuration
                </configuration>
            </execution>
          </executions>
        </plugin>
      </plugins>
  </build>

</project>

```

Spark streaming code in Java 8

```
[kafka@quickstart my-app]$ vi src/main/java/com/my

```

```
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
import org.apache.spark.streaming.api.java.*;
import org.apache.spark.streaming.kafka.*;
import org.apache.spark.streaming.Duration;
import kafka.serializer.StringDecoder;


public class SimpleSparkStreaming {

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

        Set<String> topics = Collections.singlet
        Map<String, String> kafkaParams = new Has
        kafkaParams.put("metadata.broker.list", "

        JavaPairInputDStream<String, String> dire
                String.class, String.class, Strin

        directKafkaStream.foreachRDD(rdd -> {
```

```
            System.out.println("--- New RDD with "
                    + " partitions and " + rdd.cou
            rdd.foreach(record -> System.out.prin
        });

        ssc.start();
        ssc.awaitTermination();
    }

}

```

Package with mvn

```
[kafka@quickstart my-app]$ mvn package

```

tree command to check the project structure including the generated artefacts by maven.

```
[root@quickstart my-app]# tree
.
├── dependency-reduced-pom.xml
├── nohup.out
├── pom.xml
├── src
│   ├── main
│   │   └── java
│   │       └── com
│   │           └── mycompany
│   │               └── app
│   │                   ├── App.java
│   │                   ├── KafkaConsumerExample.
│   │                   ├── KafkaProducerExample.
│   │                   └── SimpleSparkStreaming.
│   └── test
│       └── java
│           └── com
│               └── mycompany
│                   └── app
│                       └── AppTest.java
└── target
    ├── classes
    │   └── com
    │       └── mycompany
    │           └── app
    │               ├── App.class
    │               ├── KafkaConsumerExample.clas
```

```
29      |                   ├── KafkaProducerExample.class
30      |                   └── SimpleSparkStreaming.class
31      ├── generated-sources
32      |   └── annotations
33      ├── generated-test-sources
34      |   └── test-annotations
35      ├── maven-archiver
36      |   └── pom.properties
37      ├── maven-status
38      |   └── maven-compiler-plugin
39      |       ├── compile
40      |       |   └── default-compile
41      |       |       ├── createdFiles.lst
42      |       |       └── inputFiles.lst
43      |       └── testCompile
44      |           └── default-testCompile
45      |               ├── createdFiles.lst
46      |               └── inputFiles.lst
47      ├── my-app-1.0-SNAPSHOT.jar
48      ├── original-my-app-1.0-SNAPSHOT.jar
49      ├── surefire
50      ├── surefire-reports
51      |   ├── com.mycompany.app.AppTest.txt
52      |   └── TEST-com.mycompany.app.AppTest.xml
53      └── test-classes
54          └── com
55              └── mycompany
56                  └── app
57                      └── AppTest.class
58
59  33 directories, 22 files
60
```

Run the spark streaming job

Switch to root user. The password is “cloudera”.

```
1  [kafka@quickstart my-app]$ su -
2
```

Run the spark streaming job so that it continuously run to consume messages from the kafka topic “MyTestTopic”.

```
1  [root@quickstart my-app]# spark-submit   \
2  --class com.mycompany.app.SimpleSparkStreaming  \
3  --master local \
```

```
--deploy-mode client \
target/my-app-1.0-SNAPSHOT.jar

```

Kafka producer

Open a new terminal

Open a new command terminal for the producer to send messages to the kafka topic "**MyTestTopic**".

Firstly, find the container id.

```
[arul@MacBook-Pro-2:~/projects/docker-hadoop]$ docl
CONTAINER ID        IMAGE                 COMMAND
                                               NAMI
33d8cb69c173        cloudera/quickstart   "/usr/bi

```

Note down the container id: 33d8cb69c173

docker exec -it

to run a command in a running container from a separate command terminal.

```
[arul@MacBook-Pro-2:~/projects/docker-hadoop]$ docl

```

Publish messages

Publish messages to the kafka topic "**MyTestTopic**".

```
[root@quickstart /]# ./home/kafka/kafka/bin/kafka-
>

```

Start publishing messages:

```
>$$$$$$ test message sent to topic
>more messages sent
>

```

Spark streaming console

You can see the above messages consumed by the Spark stream job in the other terminal.

```
....
19/06/23 12:57:00 INFO storage.BlockManagerInfo: R
19/06/23 12:57:00 INFO spark.ContextCleaner: Clean
$$$$$$ test message sent to topic
more messages sent
19/06/23 12:57:00 INFO storage.BlockManagerInfo: R
........

```

So, you can switch between the terminals to see how the messages are produced & consumed.

‹ 27: Docker Tutorial: Apache Kafka with Java API on Cloudera quickstart

Top 10 Linux interview Q&As ›

Disclaimer