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

Posted on June 22, 2019

This requires 26: Docker Tutorial: Apache Kafka install, create topic & publish message on Cloudera quickstart. It is important that you continue from the previous tutorial.

Add maven path to ~/.bash_profile

Step 1: Maven is already in the quickstart/cloudera Docker image.

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 /]$ vi ~/.bash_profile
```

Modify so that “mvn” command can be run from any folder.

```
# .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 MAVEN_OPTS=-Dhttps.protocols=TLSv1,TLSv1.1
PATH=$PATH:/usr/local/apache-maven/apache-maven-3

```

Activate:

```
[kafka@quickstart /]$ source ~/.bash_profile

```

Now “mvn” command is available from any folder:

```
[kafka@quickstart /]$ 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: 1.7.0_67, vendor: Oracle Corporation
Java home: /usr/java/jdk1.7.0_67-cloudera/jre
Default locale: en_US, platform encoding: UTF-8
OS name: "linux", version: "4.9.125-linuxkit", arcl
[kafka@quickstart /]$

```

Create a Java project structure

Step 2: Simple Java project in maven structure.

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

```
[kafka@quickstart /]$ sudo mkdir /projects
[kafka@quickstart /]$ cd !$

```

```
[kafka@quickstart /]$ sudo chown kafka:root /proje
```

Create the maven structure:

```
[kafka@quickstart /]$ mvn archetype:generate -Dgrou
   -DartifactId=my-app \
   -DarchetypeArtifactId=maven-archetype-quickstar
   -DinteractiveMode=false

```

It takes sometime as the dependencies & plugins are downloaded.

```
[kafka@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
[kafka@quickstart projects]$

```

pom.xml

Step 3: Add kafka dependency to the pom.xml file.

```

```

```
[kafka@quickstart projects]$ cd my-app/
[kafka@quickstart my-app]$ vi pom.xml

```

Add “kafka-clients” dependency, and the plugin to build uber jars.

```
<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>

   <dependency>
    <groupId>org.apache.kafka</groupId>
    <artifactId>kafka-clients</artifactId>
    <version>0.11.0.0</version>
   </dependency>

  </dependencies>

  <!-- Build uber jar -->
  <build>
      <plugins>
        <plugin>
           <groupId>org.apache.maven.plugins</grou
           <artifactId>maven-shade-plugin</artifa
           <executions>
              <execution>
                <phase>package</phase>
                <goals>
                    <goal>shade</goal>
                </goals>
            </execution>
          </executions>
        </plugin>
      </plugins>
  </build>

```

```
</project>

```

Write the producer Java code

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

```

```
package com.mycompany.app;

import org.apache.kafka.clients.producer.*;
import org.apache.kafka.common.serialization.Long
import org.apache.kafka.common.serialization.Stri
import java.util.Properties;

public class KafkaProducerExample {
    private final static String TOPIC = "MyTestTo
    private final static String BOOTSTRAP_SERVERS
            "quickstart.cloudera:9092,quickstart.

    public static void main(String[] args) {
      Properties props = new Properties();
      props.put(ProducerConfig.BOOTSTRAP_SERVERS_
                                          BOOTSTRA
      props.put(ProducerConfig.CLIENT_ID_CONFIG, "
      props.put(ProducerConfig.KEY_SERIALIZER_CLAS
                                       LongSeriali
      props.put(ProducerConfig.VALUE_SERIALIZER_CL
                                  StringSerializer

      final Producer<Long, String> producer = new

      try {
          final ProducerRecord record =
                      new ProducerRecord(TOPIC, "He

          producer.send(record);
      } finally {
          producer.flush();
          producer.close();
      }

      System.out.println("A message has been sent

    }

}

```

Package with mvn

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

```

```
[kafka@quickstart my-app]$ tree -L 2
.
├── dependency-reduced-pom.xml
├── pom.xml
├── src
│   ├── main
│   └── test
└── target
    ├── classes
    ├── maven-archiver
    ├── my-app-1.0-SNAPSHOT.jar
    ├── original-my-app-1.0-SNAPSHOT.jar
    ├── surefire
    ├── surefire-reports
    └── test-classes

9 directories, 4 files
[kafka@quickstart my-app]$

```

Run the producer via Java code

```
[kafka@quickstart my-app]$ java -cp target/my-app-1
SLF4J: Failed to load class "org.slf4j.impl.StaticL
SLF4J: Defaulting to no-operation (NOP) logger impl
SLF4J: See http://www.slf4j.org/codes.html#StaticLo
A message has been sent to the topic: MyTestTopic
[kafka@quickstart my-app]$

```

Run the consumer

```
[kafka@quickstart my-app]$ /home/kafka/kafka/bin/ka
Hello, World
Hello, my topic
Hello From Java
Hello From Java

```

Write the consumer via Java code

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

```
package com.mycompany.app;

import org.apache.kafka.clients.consumer.*;
import org.apache.kafka.common.serialization.Longl
import org.apache.kafka.common.serialization.Strin

import java.util.Properties;
import java.util.Collections;


public class KafkaConsumerExample {
    private final static String TOPIC = "MyTestTop
    private final static String BOOTSTRAP_SERVERS
            "quickstart.cloudera:9092,quickstart.c


    public static void main(String[] args) {
      Properties props = new Properties();
      props.put(ConsumerConfig.BOOTSTRAP_SERVERS_C
                                          BOOTSTRA
      props.put(ConsumerConfig.GROUP_ID_CONFIG, "K
      props.put(ConsumerConfig.KEY_DESERIALIZER_Cl
      props.put(ConsumerConfig.VALUE_DESERIALIZER_

      final Consumer consumer = new KafkaConsumer(

      consumer.subscribe(Collections.singletonList

      int noMessageFound = 0;

      while (true) {
          ConsumerRecords consumerRecords = consum
          System.out.println("Msg count:" + consum
          if (consumerRecords.count() == 0) {
              noMessageFound++;
              if (noMessageFound > 10)
                break;
              else
                  continue;
          }

          for (Object o : consumerRecords) {
              ConsumerRecord record = (ConsumerRec
              System.out.println("Record Key " + r
              System.out.println("Record value " +
              System.out.println("Record partitio
              System.out.println("Record offset "
          }

          consumer.commitAsync();
```

```
        }

        consumer.close();

    }

}

```

Run the producer & consumer via Java code

Package:

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

```

```
[kafka@quickstart my-app]$ tree -L 7
.
├── dependency-reduced-pom.xml
├── pom.xml
├── src
│   ├── main
│   │   └── java
│   │       └── com
│   │           └── mycompany
│   │               └── app
│   │                   ├── App.java
│   │                   ├── KafkaConsumerExample.j
│   │                   └── KafkaProducerExample.j
│   └── test
│       └── java
│           └── com
│               └── mycompany
│                   └── app
│                       └── AppTest.java
└── target
    ├── classes
    │   └── com
    │       └── mycompany
    │           └── app
    │               ├── App.class
    │               ├── KafkaConsumerExample.class
    │               └── KafkaProducerExample.class
    ├── maven-archiver
    │   └── pom.properties
    ├── my-app-1.0-SNAPSHOT.jar
    ├── original-my-app-1.0-SNAPSHOT.jar
    ├── surefire
    ├── surefire-reports
```

```
    |   ├── com.mycompany.app.AppTest.txt
    |   └── TEST-com.mycompany.app.AppTest.xml
    └── test-classes
        └── com
            └── mycompany
                └── app
                    └── AppTest.class

```

Producer:

```
[kafka@quickstart my-app]$ java -cp target/my-app-1
SLF4J: Failed to load class "org.slf4j.impl.StaticL
SLF4J: Defaulting to no-operation (NOP) logger impl
SLF4J: See http://www.slf4j.org/codes.html#StaticLo
A message has been sent to the topic: MyTestTopic

```

Consumer:

```
[kafka@quickstart my-app]$ java -cp target/my-app-
SLF4J: Failed to load class "org.slf4j.impl.Static
SLF4J: Defaulting to no-operation (NOP) logger imp
SLF4J: See http://www.slf4j.org/codes.html#StaticL
Msg count:1
Record Key null
Record value Hello From Java
Record partition 0
Record offset 7
Msg count:0
Msg count:0
Msg count:0
Msg count:0
Msg count:0
Msg count:0
Msg count:0
Msg count:0
Msg count:0
Msg count:0
Msg count:0
[kafka@quickstart my-app]$

```

Getting all the messages from beginning

```
1  package com.mycompany.app;
2
3  import org.apache.kafka.clients.consumer.*;
4  import org.apache.kafka.common.serialization.Longl
5  import org.apache.kafka.common.serialization.Strir
6
7  import java.util.Properties;
8  import java.util.Collections;
9  import org.apache.kafka.common.TopicPartition;
10 import java.util.Arrays;
11 import java.util.List;
12
13 public class KafkaConsumerExample {
14     private final static String TOPIC = "MyTestTop
15     private final static String BOOTSTRAP_SERVERS
16             "quickstart.cloudera:9092,quickstart.c
17
18
19     public static void main(String[] args) {
20       Properties props = new Properties();
21       props.put(ConsumerConfig.BOOTSTRAP_SERVERS_C
22                                               BOOTSTRA
23       props.put(ConsumerConfig.GROUP_ID_CONFIG, "K
24       props.put(ConsumerConfig.KEY_DESERIALIZER_Cl
25       props.put(ConsumerConfig.VALUE_DESERIALIZER_
26       props.put("auto.offset.reset", "earliest");
27       props.put("enable.auto.commit", false);
28
29       final Consumer consumer = new KafkaConsumer(
30
31       TopicPartition topicPartition = new TopicPar
32       List<TopicPartition> partitions = Arrays.asL
33       consumer.assign(partitions);
34       consumer.seekToBeginning(partitions);
35
36       //consumer.subscribe(Collections.singletonLi
37
38       int noMessageFound = 0;
39
40       while (true) {
41           ConsumerRecords consumerRecords = consur
42           System.out.println("Msg count:" + consur
43           if (consumerRecords.count() == 0) {
44               noMessageFound++;
45               if (noMessageFound > 10)
46                 break;
47               else
48                   continue;
49           }
50
51           for (Object o : consumerRecords) {
52               ConsumerRecord record = (ConsumerRec
53               System.out.println("Record Key " + r
54               System.out.println("Record value " +
```

```
55                System.out.println("Record partitio
56                System.out.println("Record offset "
57            }
58
59            consumer.commitAsync();
60        }
61
62        consumer.close();
63
64    }
65
66 }
67
```

Compile & package it:

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

Run:

```
1 [kafka@quickstart my-app]$ java -cp target/my-app-
2
```

Output:

```
1  Msg count:8
2  Record Key null
3  Record value Hello, World
4  Record partition 0
5  Record offset 0
6  Record Key null
7  Record value Hello, my topic
8  Record partition 0
9  Record offset 1
10 Record Key null
11 Record value Hello From Java
12 Record partition 0
13 ......
14
```

‹ 26: Docker Tutorial: Apache Kafka install, create topic & publish message on Cloudera quickstart

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

Disclaimer