Ryan Ballenger
CSCIE63 Big Data Analytics
Assignment 8

Assignment 8 Solutions

Problem 1.

Problem 1) On your Cloudera VM or any other VM you might be using install Kafka. Just in case, install one of the recent Kafka 0.8 versions. Demonstrate that you can create a topic, publish messages to that topic and consume messages sent to that topic. Use Kafka command line interface.

```
// The kafka package is extracted from the downloaded .tgz file.
// Note: A second copy of Kafka is in the directory Downloads and is also used to
// demonstrate programs and output.
[cloudera@localhost Documents]$ tar -xvf kafka_2.11-0.8.2.2.tgz kafka_2.11-0.8.2.2
kafka_2.11-0.8.2.2/
kafka_2.11-0.8.2.2/LICENSE
kafka_2.11-0.8.2.2/NOTICE
kafka_2.11-0.8.2.2/bin/
kafka_2.11-0.8.2.2/bin/kafka-console-consumer.sh
kafka_2.11-0.8.2.2/bin/kafka-console-producer.sh
kafka_2.11-0.8.2.2/bin/kafka-consumer-offset-checker.sh
kafka_2.11-0.8.2.2/bin/kafka-consumer-perf-test.sh
kafka_2.11-0.8.2.2/bin/kafka-mirror-maker.sh
kafka_2.11-0.8.2.2/bin/kafka-preferred-replica-election.sh
kafka_2.11-0.8.2.2/bin/kafka-producer-perf-test.sh
kafka_2.11-0.8.2.2/bin/kafka-reassign-partitions.sh
kafka_2.11-0.8.2.2/bin/kafka-replay-log-producer.sh
kafka_2.11-0.8.2.2/bin/kafka-replica-verification.sh
kafka_2.11-0.8.2.2/bin/kafka-run-class.sh
…

// The Zookeeper Server is started with the zookeeper.properties file as a parameter
[cloudera@localhost Downloads]$ kafka_2.11-0.8.2.2/bin/zookeeper-server-start.sh
kafka_2.11-0.8.2.2/config/zookeeper.properties
[2016-04-01 18:09:36,395] INFO Reading configuration from:
kafka_2.11-0.8.2.2/config/zookeeper.properties
(org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2016-04-01 18:09:36,438] INFO autopurge.snapRetainCount set to 3
(org.apache.zookeeper.server.DatadirCleanupManager)
[2016-04-01 18:09:36,438] INFO autopurge.purgeInterval set to 0
(org.apache.zookeeper.server.DatadirCleanupManager)
```

[2016-04-01 18:09:36,438] INFO Purge task is not scheduled.
(org.apache.zookeeper.server.DatadirCleanupManager)
[2016-04-01 18:09:36,438] WARN Either no config or no quorum defined in config, running in standalone mode (org.apache.zookeeper.server.quorum.QuorumPeerMain)
[2016-04-01 18:09:36,507] INFO Reading configuration from:
kafka_2.11-0.8.2.2/config/zookeeper.properties
(org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2016-04-01 18:09:36,508] INFO Starting server
(org.apache.zookeeper.server.ZooKeeperServerMain)
[2016-04-01 18:09:36,535] INFO Server environment:zookeeper.version=3.4.6-1569965, built on 02/20/2014 09:09 GMT (org.apache.zookeeper.server.ZooKeeperServer)
…

// Next the Kafka Server is started with the server.properties file as a parameter
[cloudera@localhost Downloads]$ kafka_2.11-0.8.2.2/bin/kafka-server-start.sh
kafka_2.11-0.8.2.2/config/server.properties
...
[2016-04-01 18:15:02,850] INFO Loading offsets from [__consumer_offsets,13]
(kafka.server.OffsetManager)
[2016-04-01 18:15:02,854] INFO Finished loading offsets from [__consumer_offsets,38] in 87 milliseconds. (kafka.server.OffsetManager)
[2016-04-01 18:15:02,957] INFO Finished loading offsets from [__consumer_offsets,41] in 119 milliseconds. (kafka.server.OffsetManager)
[2016-04-01 18:15:02,957] INFO Finished loading offsets from [__consumer_offsets,3] in 110 milliseconds. (kafka.server.OffsetManager)
[2016-04-01 18:15:02,957] INFO Finished loading offsets from [__consumer_offsets,22] in 122 milliseconds. (kafka.server.OffsetManager)
[2016-04-01 18:15:02,955] INFO Finished loading offsets from [__consumer_offsets,13] in 103 milliseconds. (kafka.server.OffsetManager)
[2016-04-01 18:15:02,856] INFO Finished loading offsets from [__consumer_offsets,19] in 89 milliseconds. (kafka.server.OffsetManager)
[2016-04-01 18:15:02,958] INFO Finished loading offsets from [__consumer_offsets,32] in 3 milliseconds. (kafka.server.OffsetManager)
[2016-04-01 18:15:02,963] INFO Finished loading offsets from [__consumer_offsets,47] in 196 milliseconds. (kafka.server.OffsetManager)
[2016-04-01 18:15:02,966] INFO Finished loading offsets from [__consumer_offsets,28] in 199 milliseconds. (kafka.server.OffsetManager)
[2016-04-01 18:15:02,967] INFO Finished loading offsets from [__consumer_offsets,9] in 200 milliseconds. (kafka.server.OffsetManager)

// A topic is created named first_topic using the kafka-topics.sh file.
[cloudera@localhost Downloads]$ kafka_2.11-0.8.2.2/bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 4 --topic first_topic

Created topic "first_topic".

// The basic producer is launched with kafka-console-producer.sh on the first_topic
[cloudera@localhost Downloads]$ kafka_2.11-0.8.2.2/bin/kafka-console-producer.sh
--broker-list localhost:9092 --topic first_topic
[2016-04-01 18:21:55,557] WARN Property topic is not valid (kafka.utils.VerifiableProperties)
I'm
really
hungry!

// The basic consumer is launched. When these versions of the consumer and producer
// are both initiated, “I’m really hungery!” Is typed into the producer’s console.
[cloudera@localhost Documents]$ kafka_2.11-0.8.2.2/bin/kafka-console-consumer.sh
--zookeeper localhost:2181 --topic first_topic --from-beginning
I'm
really
hungry!

Problem 2.

Problem 2) Using Java or Python or any other (even scripting) language of your choice construct a producer and a consumer object. Let producer generate one random number between 0 or 1 and 10 every second. Let both producer and consumer run indefinitely or until you kill them. Demonstrate that your consumer is receiving messages by printing both the stream if numbers generated on the producer and the stream of numbers fetched by the consumer. You might find it easier to print to files and examine files afterwards. Once you terminate the exchange, examine Kafka’s log.
Instructions on how to write Java producer and consumer you can find on this URLs:
https://cwiki.apache.org/confluence/display/KAFKA/0.8.0+Producer+Example
https://cwiki.apache.org/confluence/display/KAFKA/0.8.0+SimpleConsumer+Example

Instructions on how to write Python clients for Kafka you could find on this URL:
https://cwiki.apache.org/confluence/display/KAFKA/Clients#Clients-Python

Instructions for Scala could be found here:
https://cwiki.apache.org/confluence/display/KAFKA/Clients#Clients-ScalaDSL

You are welcome to follow any other instructions and use any other programming or scripting language.

// Kafka-python is used on the CentOS and installation steps are traced here.
// epel-release-6-8.noarch.rpm is installed with rpm or confirmed as being installed.

```
[cloudera@localhost Downloads]$ sudo rpm -ivh
http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
Retrieving http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
Preparing...                ########################################### [100%]
        package epel-release-6-8.noarch is already installed

// python-pip is installed with yum.
[cloudera@localhost Downloads]$ sudo yum install -y python-pip
Loaded plugins: fastestmirror, refresh-packagekit, security
Setting up Install Process
Loading mirror speeds from cached hostfile
 * base: mirrors.tripadvisor.com
 * epel: mirrors.mit.edu
 * extras: mirror.us.leaseweb.net
 * updates: mirror.metrocast.net
Package python-pip-7.1.0-1.el6.noarch already installed and latest version
Nothing to do

// pip is used to install kafka-python completing the installation process
[cloudera@localhost Downloads]$ sudo pip install kafka-python
DEPRECATION: Python 2.6 is no longer supported by the Python core team, please upgrade
your Python. A future version of pip will drop support for Python 2.6
Requirement already satisfied (use --upgrade to upgrade): kafka-python in
/usr/lib/python2.6/site-packages
Requirement already satisfied (use --upgrade to upgrade): six in
/usr/lib/python2.6/site-packages (from kafka-python)


// The python producer is inputted which sends a random integer between 1 and 10
// every second
>>> from kafka import KafkaProducer
>>> import random
>>> import time
>>> producer = KafkaProducer(bootstrap_servers='localhost:9092')
>>> while True:
...     producer.send('fresh_topic', str(random.randint(1,10)))
...     time.sleep(1)
...
<kafka.producer.future.FutureRecordMetadata object at 0x17fde90>
<kafka.producer.future.FutureRecordMetadata object at 0x17fdc90>
<kafka.producer.future.FutureRecordMetadata object at 0x17fda90>
<kafka.producer.future.FutureRecordMetadata object at 0x17fd890>
<kafka.producer.future.FutureRecordMetadata object at 0x17fd690>
```

<kafka.producer.future.FutureRecordMetadata object at 0x17fd510>
<kafka.producer.future.FutureRecordMetadata object at 0x17e0cd0>
<kafka.producer.future.FutureRecordMetadata object at 0x18a2310>
<kafka.producer.future.FutureRecordMetadata object at 0x18a2110>
<kafka.producer.future.FutureRecordMetadata object at 0x18a2750>
<kafka.producer.future.FutureRecordMetadata object at 0x18a2950>
<kafka.producer.future.FutureRecordMetadata object at 0x18a2b50>
<kafka.producer.future.FutureRecordMetadata object at 0x18a2d50>
<kafka.producer.future.FutureRecordMetadata object at 0x18a2f50>
<kafka.producer.future.FutureRecordMetadata object at 0x18a3190>

// The basic producer as used in the previous problem is initiated and receives the
// random numbers.
[cloudera@localhost Documents]$ kafka_2.11-0.8.2.2/bin/kafka-console-consumer.sh
--zookeeper localhost:2181 --topic fresh_topic
2
10
5
7
10
8
6
10
2
1
1
6
8
3
7

// The logs show more detailed information about the numbers that were sent from
// the producer to the consumer in fresh_topic. The offset, position, and payload size
// specify the location and size of the sent numbers.
[cloudera@localhost Downloads]$ kafka_2.11-0.8.2.2/bin/kafka-run-class.sh
kafka.tools.DumpLogSegments --files
/tmp/kafka-logs/fresh_topic-1/00000000000000000000.log --print-data-log
Dumping /tmp/kafka-logs/fresh_topic-1/00000000000000000000.log
Starting offset: 0
offset: 0 position: 0 isvalid: true payloadsize: 1 magic: 0 compresscodec: NoCompressionCodec
crc: 2747087484 payload: 2

offset: 1 position: 27 isvalid: true payloadsize: 2 magic: 0 compresscodec: NoCompressionCodec crc: 2225123990 payload: 10
offset: 2 position: 55 isvalid: true payloadsize: 1 magic: 0 compresscodec: NoCompressionCodec crc: 1130943330 payload: 8
offset: 3 position: 82 isvalid: true payloadsize: 1 magic: 0 compresscodec: NoCompressionCodec crc: 2765160037 payload: 6
offset: 4 position: 109 isvalid: true payloadsize: 1 magic: 0 compresscodec: NoCompressionCodec crc: 2747087484 payload: 2
offset: 5 position: 136 isvalid: true payloadsize: 1 magic: 0 compresscodec: NoCompressionCodec crc: 984902598 payload: 1

Problem 3.

Problem 3) Starting from one of the attached Spark Streaming clients DirectKafkaWordCount in Java, Scala or Python write a consumer client that will replace the consumer from the previous problem. However, rather than simply printing every message it receives from the producer, let it print for us every 5 seconds the rolling count of numbers between 1 and 10 it received in the last 30 seconds. You might find it simpler to print to files and then examine those files afterwards. For Java build simple Maven Project with a single Java class and pom.xml file similar to the one provided. Build your projects following the process we used in Assignment 4.

// The edited version of direct_kafka_wordcount.py is run with spark-submit. The topic is
// specified as spark_topic. After a while, the output is pasted below. The sum of the counts is 30
// which confirms the timing is correct.
[cloudera@localhost kafka_2.11-0.8.2.2]$ spark-submit --jars libs/spark-streaming-kafka-assembly_2.11-1.6.1.jar /mnt/hgfs/VM_shared/direct_kafka_wordcount.py localhost:9092 spark_topic

...

Time: 2016-04-01 19:30:35

(u'10', 4)
(u'1', 5)
(u'3', 5)
(u'2', 3)
(u'5', 2)
(u'4', 2)
(u'7', 2)
(u'6', 3)
(u'9', 2)

(u'8', 2)

16/04/01 19:30:35 INFO WriteAheadLogManager : Attempting to clear 1 old log files in file:/home/cloudera/checkpoint/receivedBlockMetadata older than 1459564185000: file:/home/cloudera/checkpoint/receivedBlockMetadata/log-1459564121203-1459564181203
16/04/01 19:30:35 INFO WriteAheadLogManager : Cleared log files in file:/home/cloudera/checkpoint/receivedBlockMetadata older than 1459564185000
16/04/01 19:30:40 INFO utils.VerifiableProperties: Verifying properties
16/04/01 19:30:40 INFO utils.VerifiableProperties: Property group.id is overridden to
16/04/01 19:30:40 INFO utils.VerifiableProperties: Property zookeeper.connect is overridden to
16/04/01 19:30:40 INFO utils.VerifiableProperties: Verifying properties
16/04/01 19:30:40 INFO utils.VerifiableProperties: Property group.id is overridden to
16/04/01 19:30:40 INFO utils.VerifiableProperties: Property zookeeper.connect is overridden to
16/04/01 19:30:40 INFO utils.VerifiableProperties: Verifying properties
16/04/01 19:30:40 INFO utils.VerifiableProperties: Property group.id is overridden to
16/04/01 19:30:40 INFO utils.VerifiableProperties: Property zookeeper.connect is overridden to
16/04/01 19:30:40 INFO utils.VerifiableProperties: Verifying properties
16/04/01 19:30:40 INFO utils.VerifiableProperties: Property group.id is overridden to
16/04/01 19:30:40 INFO utils.VerifiableProperties: Property zookeeper.connect is overridden to
----
Time: 2016-04-01 19:30:40

(u'10', 4)
(u'1', 4)
(u'3', 5)
(u'2', 4)
(u'5', 3)
(u'4', 2)
(u'7', 2)
(u'6', 2)
(u'9', 2)
(u'8', 2)

16/04/01 19:30:40 INFO WriteAheadLogManager : Attempting to clear 0 old log files in file:/home/cloudera/checkpoint/receivedBlockMetadata older than 1459564190000:
16/04/01 19:30:40 INFO WriteAheadLogManager : Cleared log files in file:/home/cloudera/checkpoint/receivedBlockMetadata older than 1459564190000
16/04/01 19:30:45 INFO utils.VerifiableProperties: Verifying properties
16/04/01 19:30:45 INFO utils.VerifiableProperties: Property group.id is overridden to
16/04/01 19:30:45 INFO utils.VerifiableProperties: Property zookeeper.connect is overridden to
16/04/01 19:30:45 INFO utils.VerifiableProperties: Verifying properties
16/04/01 19:30:45 INFO utils.VerifiableProperties: Property group.id is overridden to
16/04/01 19:30:45 INFO utils.VerifiableProperties: Property zookeeper.connect is overridden to

```
16/04/01 19:30:45 INFO utils.VerifiableProperties: Verifying properties
16/04/01 19:30:45 INFO utils.VerifiableProperties: Property group.id is overridden to
16/04/01 19:30:45 INFO utils.VerifiableProperties: Property zookeeper.connect is overridden to
16/04/01 19:30:45 INFO utils.VerifiableProperties: Verifying properties
16/04/01 19:30:45 INFO utils.VerifiableProperties: Property group.id is overridden to
16/04/01 19:30:45 INFO utils.VerifiableProperties: Property zookeeper.connect is overridden to
-------------------------------------------
Time: 2016-04-01 19:30:45
-------------------------------------------
(u'10', 4)
(u'1', 2)
(u'3', 5)
(u'2', 5)
(u'5', 3)
(u'4', 4)
(u'7', 1)
(u'6', 2)
(u'9', 2)
(u'8', 2)

…

// The same python producer from problem 2 is run. An integer from 1 to 10 is sent
// every second.
>>> from kafka import KafkaProducer
>>> import random
>>> import time
>>>
>>> producer = KafkaProducer(bootstrap_servers='localhost:9092')
>>> while True:
...     producer.send('spark_topic', str(random.randint(1,10)))
...     time.sleep(1)
...
<kafka.producer.future.FutureRecordMetadata object at 0xb164d0>
<kafka.producer.future.FutureRecordMetadata object at 0xb16690>
<kafka.producer.future.FutureRecordMetadata object at 0xb16890>
<kafka.producer.future.FutureRecordMetadata object at 0xb16a90>
<kafka.producer.future.FutureRecordMetadata object at 0xb16c90>
<kafka.producer.future.FutureRecordMetadata object at 0xb16e90>
<kafka.producer.future.FutureRecordMetadata object at 0xbb80d0>
<kafka.producer.future.FutureRecordMetadata object at 0xbb82d0>
<kafka.producer.future.FutureRecordMetadata object at 0xbb84d0>
<kafka.producer.future.FutureRecordMetadata object at 0xbb86d0>
```

<kafka.producer.future.FutureRecordMetadata object at 0xb16ed0>
<kafka.producer.future.FutureRecordMetadata object at 0xb16cd0>
<kafka.producer.future.FutureRecordMetadata object at 0xb16ad0>
<kafka.producer.future.FutureRecordMetadata object at 0xb168d0>
<kafka.producer.future.FutureRecordMetadata object at 0xb166d0>
<kafka.producer.future.FutureRecordMetadata object at 0xb16550>
<kafka.producer.future.FutureRecordMetadata object at 0xbb8750>
<kafka.producer.future.FutureRecordMetadata object at 0xbb8590>
<kafka.producer.future.FutureRecordMetadata object at 0xbb8390>
<kafka.producer.future.FutureRecordMetadata object at 0xbb8190>
<kafka.producer.future.FutureRecordMetadata object at 0xbb89d0>
<kafka.producer.future.FutureRecordMetadata object at 0xbb8bd0>
<kafka.producer.future.FutureRecordMetadata object at 0xbb8dd0>
…

direct_kakfa_wordcount.py

```
"""
 Counts words in UTF8 encoded, '\n' delimited text directly received from Kafka in every 2
seconds.
 Usage: direct_kafka_wordcount.py <broker_list> <topic>

 To run this on your local machine, you need to setup Kafka and create a producer first, see
 http://kafka.apache.org/documentation.html#quickstart

 and then run the example
    `$ bin/spark-submit --jars \
      external/kafka-assembly/target/scala-*/spark-streaming-kafka-assembly-*.jar \
      examples/src/main/python/streaming/direct_kafka_wordcount.py \
      localhost:9092 test`
"""

import sys
import Queue

from pyspark import SparkContext
from pyspark.streaming import StreamingContext
from pyspark.streaming.kafka import KafkaUtils
```

```
if __name__ == "__main__":
    if len(sys.argv) != 3:
        print("Usage: direct_kafka_wordcount.py <broker_list> <topic>", sys.stderr)
        exit(-1)


    sc = SparkContext(appName="PythonStreamingDirectKafkaWordCount")

    # reduces logging significantly
    logging = sc._jvm.org.apache.log4j
    logging.LogManager.getLogger("org"). setLevel( logger.Level.ERROR )
    logging.LogManager.getLogger("akka").setLevel( logger.Level.ERROR )

    ssc = StreamingContext(sc, 5)

    ssc.checkpoint("file:///home/cloudera/checkpoint")

    brokers, topic = sys.argv[1:]
    kvs = KafkaUtils.createDirectStream(ssc, [topic], {"metadata.broker.list": brokers})

    lines = kvs.map(lambda xtor: xtor[1])

    # added reduce by key and window
    counts = lines.flatMap(lambda linetor: linetor.split(" ")) \
        .map(lambda wordtor: (wordtor, 1)) \
        .reduceByKeyAndWindow((lambda ator, btor: ator+btor), (lambda ator, btor: ator-btor),
30,5)
    counts.pprint()

    ssc.start()
    ssc.awaitTermination()
```