Assignment 9
CSCI E 63 - Big Data Analytics

Problem 1:
Install Kafka on you Linux VM. If on your own VM with CentOS7.4 you should be able to install Kafka using yum:

```
$ sudo yum install kafka
```

Kafka code is most probably installed in the directory `/usr/hdp/current/kafka-broker`. Create an environmental variable KAFKA_HOME pointing to that directory. Place the directory `/usr/hdp/current/kafka-broker/bin` in the PATH variable in the `.bash_profile` file in your home directory. Source `.bash_profile` (e.i. issue command `$ source .bash_profile`), so that you can invoke Kafka scripts from any directory. Make sure that Zookeeper server is started. Kafka configuration files reside in the directories: `$KAFKA_HOME/config`. Create a topic. Demonstrate that provided scripting client `kafka-console-producer.sh` receives and displays messages produced by `kafka-console-consumer.sh` client.

Answer:

→ Install Kafka on the linux VM.

Download Apache Kafka from the website

```
cd ~
wget http://www-us.apache.org/dist/kafka/0.11.0.1/kafka_2.12-0.11.0.1.tgz
```

Unzip the archive to a preferred location, such as /opt:

```
tar -xvf kafka_2.12-0.11.0.1.tgz
sudo mv kafka_2.12-0.11.0.1 /opt
```

Create an environment variable KAFKA_HOME

```
cd ~
vi .bash_profile
```

Add the lines:

```
export KAFKA_HOME=/opt/kafka_2.12-0.11.0.1
PATH=$PATH:$KAFKA_HOME/bin
```

Build the profile using command:

```
source .bash_profile
```

Assignment 9
CSCI E 63 - Big Data Analytics

Start the Zookeeper server

```
cd $KAFKA_HOME
bin/zookeeper-server-start.sh -daemon config/zookeeper.properties
```

Start the Kafka server

```
bin/kafka-server-start.sh config/server.properties
```

```
[2017-11-04 08:15:28,510] INFO Kafka version : 0.11.0.1 (org.apache.kafka.common.utils.AppInfoParser)
[2017-11-04 08:15:28,510] INFO Kafka commitId : c2a0d5f9b1f45bf5 (org.apache.kafka.common.utils.AppInfoParser)
[2017-11-04 08:15:28,511] INFO [Kafka Server 0], started (kafka.server.KafkaServer)
```

Create a new topic "test" in a new SSH connection

```
cd $KAFKA_HOME
bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --
partitions 1 --topic test
```

```
[kbhandarkar@localhost kafka_2.12-0.11.0.1]$ bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partiti
ons 1 --topic test
OpenJDK 64-Bit Server VM warning: If the number of processors is expected to increase from one, then you should configure the number
of parallel GC threads appropriately using -XX:ParallelGCThreads=N
Created topic "test".
```

Producer Terminal

```
cd $KAFKA_HOME
bin/kafka-console-producer.sh --broker-list localhost:9092 --topic test
```

Consumer Terminal

```
cd $KAFKA_HOME
bin/kafka-console-consumer.sh --zookeeper localhost:2181 --topic test --from-
beginning
```

Demonstration

```
[kbhandarkar@localhost ~]$ cd $KAFKA_HOME
[kbhandarkar@localhost kafka_2.12-0.11.0.1]$ bin/kafka-console-producer.sh --broker-
list localhost:9092 --topic test
OpenJDK 64-Bit Server VM warning: If the number of processors is expected to increas
e from one, then you should configure the number of parallel GC threads appropriatel
y using -XX:ParallelGCThreads=N
>Sending
>some
>messages
>
```

```
[kbhandarkar@localhost ~]$ cd $KAFKA_HOME
[kbhandarkar@localhost kafka_2.12-0.11.0.1]$ bin/kafka-topics.sh --create --zookeepe
r localhost:2181 --replication-factor 1 --partitions 1 --topic test
OpenJDK 64-Bit Server VM warning: If the number of processors is expected to increas
e from one, then you should configure the number of parallel GC threads appropriatel
y using -XX:ParallelGCThreads=N
Created topic "test".
[kbhandarkar@localhost kafka_2.12-0.11.0.1]$ bin/kafka-console-consumer.sh --zookeep
er localhost:2181 --topic test --from-beginning
OpenJDK 64-Bit Server VM warning: If the number of processors is expected to increas
e from one, then you should configure the number of parallel GC threads appropriatel
y using -XX:ParallelGCThreads=N
Using the ConsoleConsumer with old consumer is deprecated and will be removed in a f
uture major release. Consider using the new consumer by passing [bootstrap-server] i
nstead of [zookeeper].
Sending
some
messages
```

Karan A. Bhandarkar

Assignment 9
CSCI E 63 - Big Data Analytics

Problem 2:
Make supplied python script `kafka_consumer.py` receive messages produced by supplied python script `kafka_producer.py`. Modify `kafka_producer.py` so that you can pass server name and the port of the Kafka broker and the name of Kafka topic on the command line. Also, modify that script so that it continuously reads your terminal inputs and sends every line to Kafka consumer. Demonstrate that kafka_consumer.py can read and display messages of modified `kafka_producer.py`. Provide working code of modified `kafka_producer.py`. Describe to us the process of installing Python packages, if any, you needed for this problem.

Answer:

→ Instal Kafka Python

```
Sudo pip install kafka-python
```

```
[kbhandarkar@localhost kafka_2.12-0.11.0.1]$ sudo pip install kafka-python
[sudo] password for kbhandarkar:
Collecting kafka-python
  Downloading kafka_python-1.3.5-py2.py3-none-any.whl (207kB)
    100% |████████████████████████████████| 215kB 1.7MB/s
Installing collected packages: kafka-python
Successfully installed kafka-python-1.3.5
```

→ Modify `kafka_producer.py` so that you can pass server name and the port of the Kafka broker and the name of Kafka topic on the command line.

Change lines:
producer = KafkaProducer(bootstrap_servers='localhost:9092')
topic = 'test3'

To:
producer = KafkaProducer(bootstrap_servers=sys.argv[1])
topic = sys.argv[2]

→ Also, modify that script so that it continuously reads your terminal inputs and sends every line to Kafka consumer.

Change script logic to:

```
while 1:
    try:
        line =sys.stdin.readline()
```

```
except  KeyboardInterrupt:
        break
if not line:
        break
print ("sending Message",line)
producer.send(topic,line)
```

Complete script submitted as `kafka_producer_p2.py`

→ Demonstrate that kafka_consumer.py can read and display messages of modified `kafka_producer.py`.

Start the producer

```
python kafka_producer_p2.py localhost:9092 test
```

```
[kbhandarkar@localhost Assignment9]$ python kafka_producer_p2.py localhost:9092 test
Running script on:
('Server: ', 'localhost:9092')
('Topic: ', 'test')
```

Start the consumer

```
python kafka_consumer.py localhost:9092 test
```

```
INFO:kafka.cluster:Group coordinator for my-group1 is BrokerMetadata(nodeId=0, host=
u'localhost', port=9092, rack=None)
INFO:kafka.coordinator:Discovered coordinator 0 for group my-group1
INFO:kafka.coordinator:(Re-)joining group my-group1
INFO:kafka.coordinator:Joined group 'my-group1' (generation 1) with member_id kafka-
python-1.3.5-ea5d4ec1-c7bb-4438-bfbb-9fa825a37924
INFO:kafka.coordinator:Elected group leader -- performing partition assignments usin
g range
INFO:kafka.coordinator:Successfully joined group my-group1 with generation 1
INFO:kafka.consumer.subscription_state:Updated partition assignment: [TopicPartition
(topic=u'test', partition=0)]
INFO:kafka.coordinator.consumer:Setting newly assigned partitions set([TopicPartitio
n(topic=u'test', partition=0)]) for group my-group1
```

Test

```
[kbhandarkar@localhost Assignment9]$ python kafka_producer_p2.py localhost:9092 test
Running script on:
('Server: ', 'localhost:9092')
('Topic: ', 'test')
does
('sending Message', 'does\n')
this
('sending Message', 'this\n')
keep
('sending Message', 'keep\n')
sending?
('sending Message', 'sending?\n')
```

```
got msg:  ConsumerRecord(topic=u'test', partition=0, offset=3, timestamp=15098042519
88, timestamp_type=0, key=None, value='does\n', checksum=-1830256854, serialized_key
_size=-1, serialized_value_size=5)
partition:  0 message offset:  3
got msg:  ConsumerRecord(topic=u'test', partition=0, offset=4, timestamp=15098042567
52, timestamp_type=0, key=None, value='this\n', checksum=-551336841, serialized_key_
size=-1, serialized_value_size=5)
partition:  0 message offset:  4
got msg:  ConsumerRecord(topic=u'test', partition=0, offset=5, timestamp=15098042587
86, timestamp_type=0, key=None, value='keep\n', checksum=-503734953, serialized_key_
size=-1, serialized_value_size=5)
partition:  0 message offset:  5
got msg:  ConsumerRecord(topic=u'test', partition=0, offset=6, timestamp=15098042616
57, timestamp_type=0, key=None, value='sending?\n', checksum=-53976713, serialized_k
ey_size=-1, serialized_value_size=9)
partition:  0 message offset:  6
```

Assignment 9
CSCI E 63 - Big Data Analytics

Problem 3:
Rather than using `splitAndSend.sh` bash script to generate traffic towards Spark Streaming engine, write a Kafka Producer which will read `orders.txt` file and send 1,000 orders to a Kafka topic every second. Write a Kafka consumer that will deliver those batches of orders to Spark Streaming engine. Base your Kafka consumer on provided `direct_word_count.py` script. Let Spark streaming engine count the number of orders different stocks where bought in each batch. Display for us a section of results in your solution. Describe to us the process of installing and invoking Python packages, if any, you needed for this problem.

Answer:

→ Download jar from maven central repository

```
-rw-rw-r--. 1 kbhandarkar kbhandarkar 11043745 Nov  4 11:09 spark-streaming-kafka-0-
8-assembly 2.10-2.2.0.jar
```

→ Producer code snippet(full script submitted separately)

```
f = open("orders.txt","r")

lineList = []
for line in f:
        lineList.append(line)
        if (len(lineList)%999) == 0:
                for i,x in enumerate(lineList):
                        producer.send(topic,lineList[i])
                time.sleep(1)
        print ("Sent batch #", len(lineList)/999)
```

→ Start Consumer

```
$SPARK_HOME/bin/spark-submit --jars spark-streaming-kafka-0-8-
assembly_2.10-2.2.0.jar direct_word_count.py localhost:9092 test
```

→ Start Producer

```
python kafka_producer_p3.py localhost:9092 test
```

Assignment 9
CSCI E 63 - Big Data Analytics

→ Display a section of the results

```
Time: 2017-10-28 11:31:52
-------------------------------------------
(u'20:25:28,6737,76,INO,24,31.00,S\n', 1)
(u'20:25:28,18928,40,HMY,356,12.00,B\n', 1)
(u'20:25:28,899,92,GG,395,94.00,S\n', 1)
(u'20:25:28,17161,78,PYPL,388,60.00,S\n', 1)
(u'20:25:28,11433,21,PBR,440,27.00,S\n', 1)
(u'20:25:28,8085,64,NQ,396,91.00,S\n', 1)
(u'20:25:28,15550,67,CIG,717,15.00,B\n', 1)
(u'20:25:28,18549,54,X,167,66.00,S\n', 1)
(u'20:25:28,5854,22,SDLP,294,18.00,B\n', 1)
(u'20:25:28,7484,101,AUY,639,62.00,S\n', 1)
...
```

Karan A. Bhandarkar

Assignment 9
CSCI E 63 - Big Data Analytics

Problem 4:
nstall Cassandra server on your VM. Use Cassandra SQL Client, `cqlsh`, to create and populate table `person`. Let every `person` by described by his or her first and last name, and city where he or she lives. Let every person possess up to three cell phones. Populate your table with three individuals using `cqlsh` client. Demonstrate that you can select the content of your table `person` including individuals' cell phones. Write a simple client in a language of your choice that will populate 3 rows in Casandra's table `person`, subsequently update one of those rows, for example change the city where a person lives, and finally retrieve that modify row from Cassandra and write its content to the console. Describe to us the process of installing and invoking Java, Scala or Python packages, if any, you needed for this problem.

Answer:

→ Install Casandra
Create file /etc/yum.repos.d/datastax.repo referring to the DataStax Community repository. Add the following lines to the file:
[datastax]
name = DataStax Repo for Apache Cassandra
baseurl = https://rpm.datastax.com/community
enabled = 1
gpgcheck = 0

Install the packages:
```
sudo yum install dsc21-2.1.13-1 cassandra2.1.6-1
sudo yum install cassandra21-tools-2.1.13-1 ##Installs optional utilities.
```

→ Verify and start Cassandra

```
[cassandra@quickstart root]$ which cassandra
/usr/sbin/cassandra
```

```
cd /usr/sbin
cassandra -f
```

On successful start, end of console looks like:

```
INFO  19:43:09 Starting listening for CQL clients on localhost/127.0.0.1:9042...
INFO  19:43:10 Binding thrift service to localhost/127.0.0.1:9160
INFO  19:43:10 Listening for thrift clients...
```

Assignment 9
CSCI E 63 - Big Data Analytics

→ Invoke cqlsh

```
[kbhandarkar@localhost ~]$ cqlsh
Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 2.1.13 | CQL spec 3.2.1 | Native protocol v3]
Use HELP for help.
cqlsh>
```

→ Create tables and insert rows

```
cqlsh> CREATE KEYSPACE mykeyspace WITH REPLICATION= {'class' :'SimpleStrategy',
'replication_factor' :1};

cqlsh> use mykeyspace;
cqlsh:mykeyspace> CREATE TABLE person (person_id int PRIMARY KEY, fname text, lname
text, city text,cellphone1 text, cellphone2 text, cellphone3 text);
cqlsh:mykeyspace> desc person;

CREATE TABLE mykeyspace.person (
    person_id int PRIMARY KEY,
    cellphone1 text,
    cellphone2 text,
    cellphone3 text,
    city text,
    fname text,
    lname text
) WITH bloom_filter_fp_chance = 0.01
    AND caching = '{"keys":"ALL", "rows_per_partition":"NONE"}'
    AND comment = ''
    AND compaction = {'class':
'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy'}
    AND compression = {'sstable_compression':
'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND dclocal_read_repair_chance = 0.1
    AND default_time_to_live = 0
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair_chance = 0.0
    AND speculative_retry = '99.0PERCENTILE';

cqlsh:mykeyspace> insert into person
(person_id ,fname,lname ,city,cellphone1,cellphone2,cellphone3) VALUES
(1,'Brian','Smith','Cleveland','123-111-1456' ,'342-987-4532','367-544-6666' ) ;
cqlsh:mykeyspace> select * from person;

 person_id | cellphone1   | cellphone2   | cellphone3   | city      | fname | lname
-----------+--------------+--------------+--------------+-----------+-------+-------
         1 | 123-111-1456 | 342-987-4532 | 367-544-6666 | Cleveland | Brian | Smith

(1 rows)
```

Karan A. Bhandarkar

Assignment 9
CSCI E 63 - Big Data Analytics

```
cqlsh:mykeyspace> insert into person (person_id ,fname,lname
              ... ,city,cellphone1,cellphone2,cellphone3) VALUES
(2,'John','Doe','Newyork','123-111-1131'
              ... ,'342-222-4532','367-333-8786' ) ;
cqlsh:mykeyspace> select * from person ;
```

```
 person_id | cellphone1   | cellphone2   | cellphone3   | city      | fname | lname
-----------+--------------+--------------+--------------+-----------+-------+-------
         1 | 123-111-1456 | 342-987-4532 | 367-544-6666 | Cleveland | Brian | Smith
         2 | 123-111-1131 | 342-222-4532 | 367-333-8786 |   Newyork |  John |   Doe
```

```
cqlsh:mykeyspace> insert into person
(person_id ,fname,lname,city,cellphone1,cellphone2,cellphone3) VALUES(3,
'Karan','Bhandarkar','Cambridge','860-990-7435','111-111-1111','222-222-2222');
cqlsh:mykeyspace> select * from person;
```

```
 person_id | cellphone1   | cellphone2   | cellphone3   | city      | fname | lname
-----------+--------------+--------------+--------------+-----------+-------+-----------
         1 | 123-111-1456 | 342-987-4532 | 367-544-6666 | Cleveland | Brian |     Smith
         2 | 123-111-1131 | 342-222-4532 | 367-333-8786 |   Newyork |  John |       Doe
         3 | 860-990-7435 | 111-111-1111 | 222-222-2222 | Cambridge | Karan | Bhandarkar
```

→ Update one row

```
cqlsh:mykeyspace> update person set city ='San Fransisco' where person_id = 2;
cqlsh:mykeyspace> select * from person;
```

```
 person_id | cellphone1   | cellphone2   | cellphone3   | city          | fname | lname
-----------+--------------+--------------+--------------+---------------+-------+-----------
         1 | 123-111-1456 | 342-987-4532 | 367-544-6666 |     Cleveland | Brian |     Smith
         2 | 123-111-1131 | 342-222-4532 | 367-333-8786 | San Fransisco |  John |       Doe
         3 | 860-990-7435 | 111-111-1111 | 222-222-2222 |     Cambridge | Karan | Bhandarkar
```

→ Install Cassandra driver

```
sudo pip install cassandra-driver
```

→ Insert rows using script cassandra_p4_1.py(submitted separately)

```
python cassandra_p4_1.py
```

```
[kbhandarkar@localhost Assignment9]$ python cassandra_p4_1.py
Insert Done
```

Karan A. Bhandarkar

Assignment 9
CSCI E 63 - Big Data Analytics

Verify update

```
cqlsh:mykeyspace> select * from person;
```

```
 person_id | cellphone1   | cellphone2   | cellphone3   | city          | fname | lname
-----------+--------------+--------------+--------------+---------------+-------+------------
         5 | 508-767-1111 | 508-222-4535 | 508-333-3333 |       Buffalo | Smith |     Duncun
         1 | 123-111-1456 | 342-987-4532 | 367-544-6666 |     Cleveland | Brian |      Smith
         2 | 123-111-1131 | 342-222-4532 | 367-333-8786 | San Fransisco |  John |        Doe
         4 | 508-111-1111 | 508-222-2222 | 508-333-3333 |        Boston |   Tom |    Bowmore
         6 | 508-617-1511 | 508-222-2222 | 508-333-3333 |     Cleveland |   Sam |    Richard
         3 | 860-990-7435 | 111-111-1111 | 222-222-2222 |     Cambridge | Karan | Bhandarkar

(6 rows)
```

→ Update rows using script cassandra_p4_1.py(submitted separately)

```
python cassandra_p4_2.py
```

```
[kbhandarkar@localhost Assignment9]$ python cassandra_p4_2.py
 Now Doing Update
('First Name=', u'Smith', 'Last Name =', u'Duncun', 'City =', u'Erie')
```

Verify update

```
cqlsh:mykeyspace> select * from person;
```

```
 person_id | cellphone1   | cellphone2   | cellphone3   | city          | fname | lname
-----------+--------------+--------------+--------------+---------------+-------+------------
         5 | 508-767-1111 | 508-222-4535 | 508-333-3333 |          Erie | Smith |     Duncun
         1 | 123-111-1456 | 342-987-4532 | 367-544-6666 |     Cleveland | Brian |      Smith
         2 | 123-111-1131 | 342-222-4532 | 367-333-8786 | San Fransisco |  John |        Doe
         4 | 508-111-1111 | 508-222-2222 | 508-333-3333 |        Boston |   Tom |    Bowmore
         6 | 508-617-1511 | 508-222-2222 | 508-333-3333 |     Cleveland |   Sam |    Richard
         3 | 860-990-7435 | 111-111-1111 | 222-222-2222 |     Cambridge | Karan | Bhandarkar

(6 rows)
```