Rishabh Jain
CSP-554
A20495530

Assignment - 9

Q1) What is the Kappa architecture and how does it differ from the lambda architecture?

Sol) As per the article in the kappa architecture, precomputation of data does not happen on regular basis in the batch layer. It is the stream processing system in which all the computation is done. The re-computation is performed only when the business logic changes by replaying past data. For this, the Kappa Architecture uses powerful stream processor which can deal with data at faster speed than the data comes in.

Whereas Lambda architecture is a system consisting of three layers: Batch, Speed and Serving layer. It targets both Volume and Velocity challenge of big data at the same time. It has both batch-oriented system and real-time system.

Q2) What are the advantages and drawbacks of pure streaming versus micro-batch real-time processing systems?

Sol) Storm and Samza are pure stream-oriented systems with very low latency and somewhat high per-item costs, whereas batch-oriented systems offer unmatched resource efficiency at the tradeoff of unreasonably high latency for real-time applications.

Data is buffered and processed in batches in micro-batch real-time processing systems. It improves efficiency while also lengthening the time an individual item spends in the data flow. Storm Trident and Spark Streaming are two examples of this type.

Q3) In few sentences describe the data processing pipeline in Storm.

Sol) A topology in Storm refers to a data pipeline or application. Spouts are the nodes that intake data and so start the data flow in the topology. Spouts output tuples to bolts, which execute processing, write data to external storage, and may transmit tuples further downstream. Data flow between nodes is controlled by storm groupings. Storm distributes spouts and bolts in a round-robin fashion by default, but the scheduler can be modified to accommodate for cases in which a specific processing step must be performed on a specific node.

Storm, on the other hand, does not ensure the sequence in which tuples are processed; however, it does offer the option of at-least-once processing via an acknowledgement feature that maintains the processing status of every single tuple as it travels through the topology.

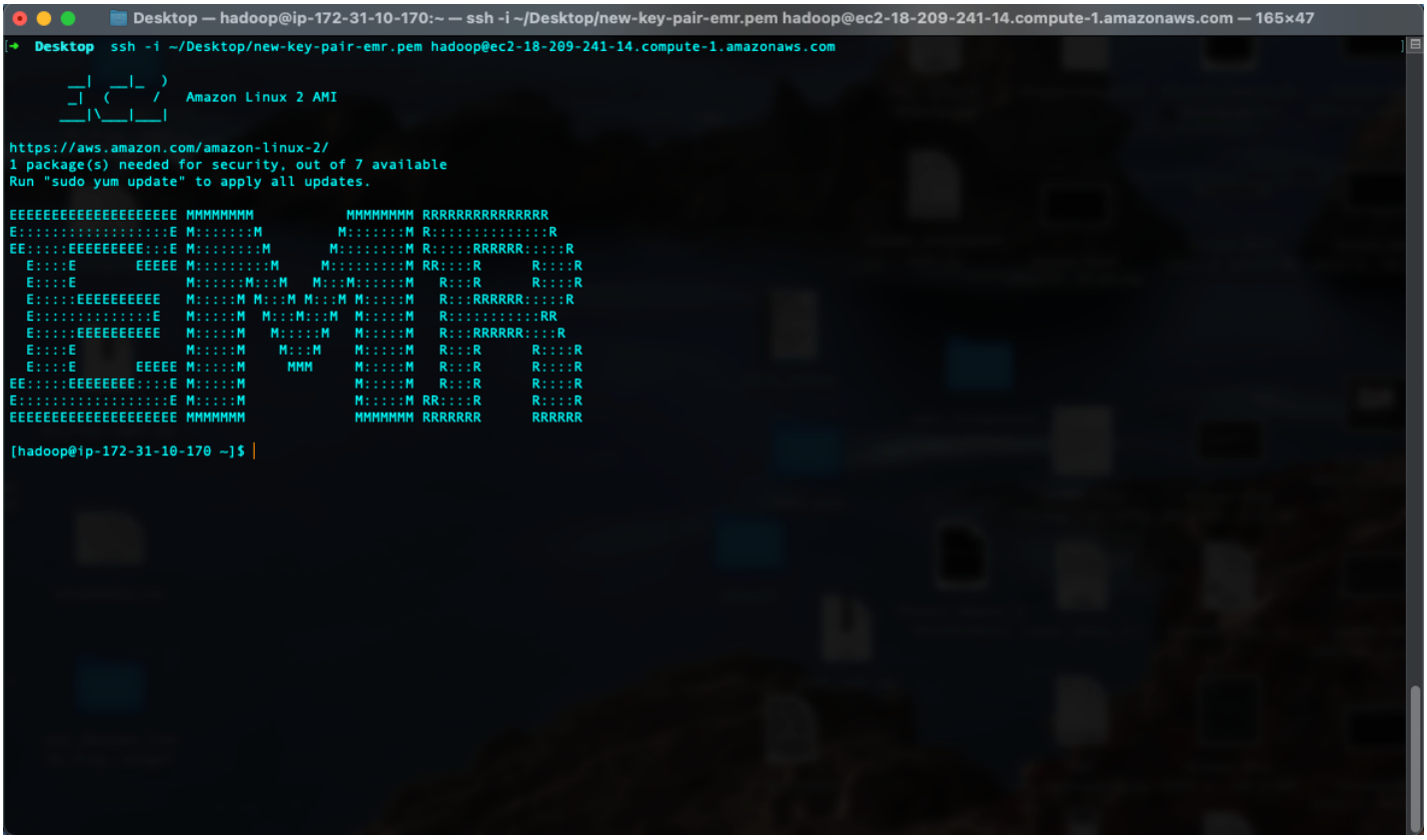
Q4) How does Spark streaming shift the Spark batch processing approach to work on real-time data streams?

Sol) By chunking the stream of incoming data items into small batches, translating them into DDs, and processing them as usual, Spark streaming shifts the batch-processing method towards real-time requirements. It also automatically manages data flow and distribution.

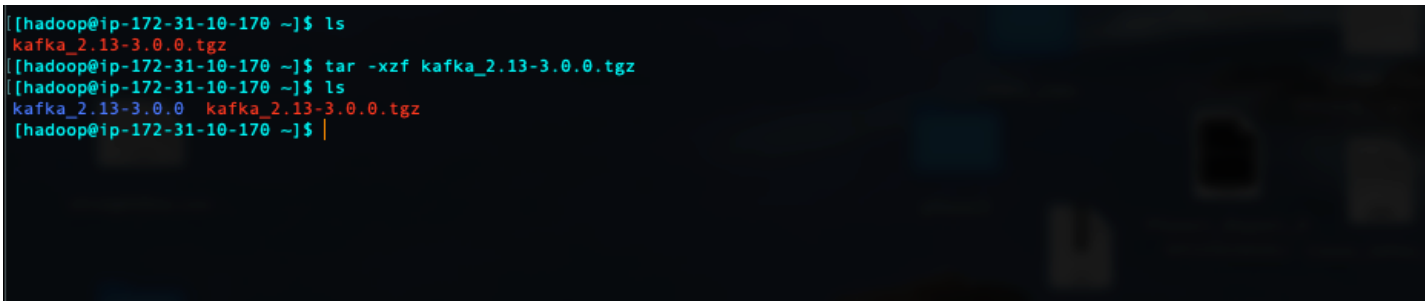
Rishabh Jain
CSP-554
A20495530

Extra Credit exercise

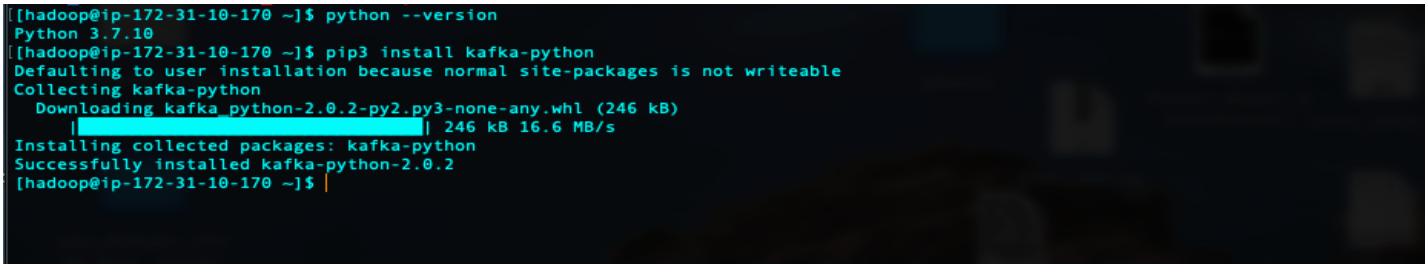
Starting a EMR cluster



Extracting the Kafka Package

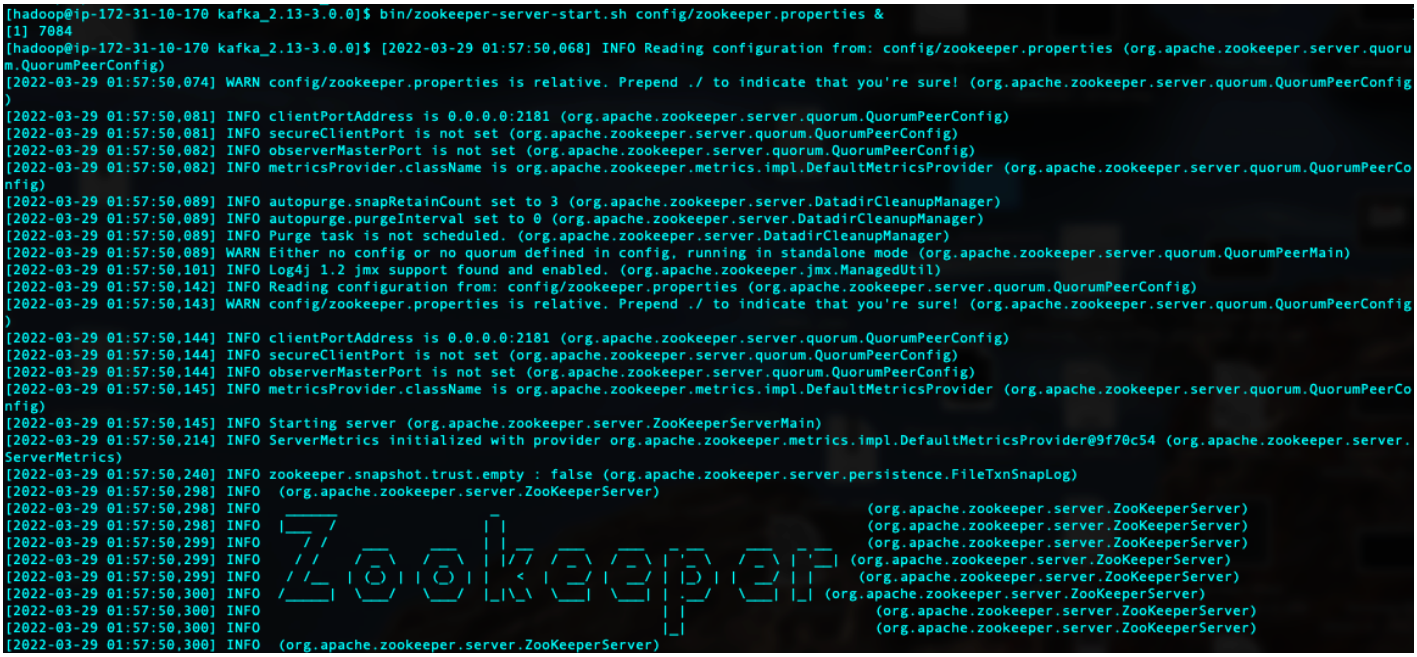


Installing the kafka-python package

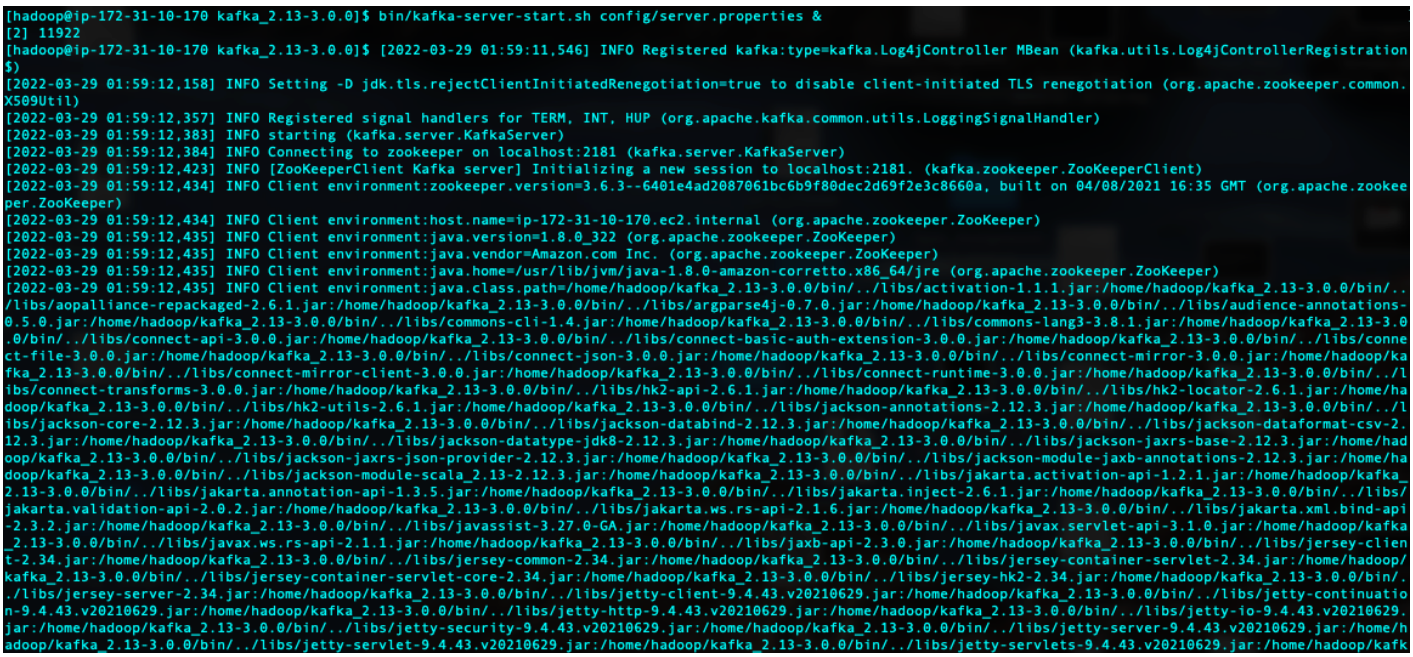


Rishabh Jain
CSP-554
A20495530

Running command: - bin/zookeeper-server-start.sh config/zookeeper.properties &



Running command: - bin/kafka-server-start.sh config/server.properties &



Rishabh Jain
CSP-554
A20495530

At Producer-terminal
Creating a topic named “sample”
Running command: - bin/kafka-topics.sh --create --replication-factor 1 --partitions 1 --bootstrap-server localhost:9092 --topic sample

```
[➜  Desktop  ssh -i ~/Desktop/new-key-pair-emr.pem hadoop@ec2-18-209-241-14.compute-1.amazonaws.com

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
1 package(s) needed for security, out of 7 available
Run "sudo yum update" to apply all updates.

EEEEEEEEEEEEEEEEEEE MMMMMMMM           MMMMMMMM RRRRRRRRRRRRRRR
E::::::::::::::::::E M::::::M         M::::::M R::::::::::::::R
EE:::::EEEEEEEEE:::E M::::::::M       M::::::::M R:::::RRRRRR:::::R
  E::::E       EEEEE M:::::::::M     M:::::::::M RR::::R      R::::R
  E::::E             M::::::M:::M   M:::M::::::M   R:::R      R::::R
  E:::::EEEEEEEEEE   M:::::M M:::M M:::M M:::::M   R:::RRRRRR:::::R
  E::::::::::::::E   M:::::M  M:::M:::M  M:::::M   R:::::::::::RR
  E:::::EEEEEEEEEE   M:::::M   M:::::M   M:::::M   R:::RRRRRR:::R
  E::::E             M:::::M    M:::M    M:::::M   R:::R      R::::R
  E::::E       EEEEE M:::::M     MMM     M:::::M   R:::R      R::::R
EE:::::EEEEEEEE:::E M:::::M             M:::::M   R:::R      R::::R
E::::::::::::::::::E M:::::M             M:::::M RR::::R      R::::R
EEEEEEEEEEEEEEEEEEE MMMMMMM             MMMMMMM RRRRRRR      RRRRRR

[[hadoop@ip-172-31-10-170 ~]$ cd kafka_2.13-3.0.0
[hadoop@ip-172-31-10-170 kafka_2.13-3.0.0]$ bin/kafka-topics.sh --create --replication-factor 1 --partitions 1 --bootstrap-server
localhost:9092 --topic sample

Created topic sample.
[hadoop@ip-172-31-10-170 kafka_2.13-3.0.0]$
[hadoop@ip-172-31-10-170 kafka_2.13-3.0.0]$
```

More topics created

```
Desktop — hadoop@ip-172-31-10-170:~/kafka_2.13-3.0.0 — ssh -i ~/Desktop/new-key-pair-emr.pem hadoop@ec2-18-209-241-14....
[[hadoop@ip-172-31-10-170 kafka_2.13-3.0.0]$ bin/kafka-topics.sh --create --replication-factor 1 --partitions 1 --bootstrap-server ]
localhost:9092 --topic rishabh_jain
WARNING: Due to limitations in metric names, topics with a period ('.') or underscore ('_') could collide. To avoid issues it is b
est to use either, but not both.
Created topic rishabh_jain.
[[hadoop@ip-172-31-10-170 kafka_2.13-3.0.0]$ bin/kafka-topics.sh --create --replication-factor 1 --partitions 1 --bootstrap-server ]
localhost:9092 --topic rishabh507
Created topic rishabh507.
[hadoop@ip-172-31-10-170 kafka_2.13-3.0.0]$
```

Rishabh Jain
CSP-554
A20495530

Listing all topics

```
[hadoop@ip-172-31-10-170 kafka_2.13-3.0.0]$ bin/kafka-topics.sh --list --bootstrap-server localhost:9092
rishabh507
rishabh_jain
sample
[hadoop@ip-172-31-10-170 kafka_2.13-3.0.0]$
```

On producer terminal
vim put.py (code file attached with the submission on blackboard).

```
1  from time import sleep
2  from json import dumps
3  from kafka import KafkaProducer
4
5  Producer = KafkaProducer(bootstrap_servers = ['localhost:9092'],value_serializer = lambda x: dumps(x).encode('utf-8'))
6
7  synmyid = 'MYID'
8  synmyname = 'MYNAME'
9  synmyeyecolor = 'MYEYECOLOR'
10
11 realid = input("Enter your ID: ")
12 realname = input("Enter your name: ")
13 realeyecolor = input("Enter your eye color: ")
14
15 my_dict = {}
16 my_dict[synmyid] = realid
17
18 my_dict1 = {}
19 my_dict1[synmyname] = realname
20
21 my_dict2 = {}
22 my_dict2[synmyeyecolor] = realeyecolor
23
24 myid = my_dict
25 Producer.send('sample',myid)
26 sleep(4)
27
28 myname = my_dict1
29 Producer.send('sample',myname)
30 sleep(4)
31
32 myeyecolor = my_dict2
33 Producer.send('sample',myeyecolor)
34 sleep(4)
35
36 Producer.close()
```

Python3 put.py

```
Desktop — hadoop@ip-172-31-15-220:~/kafka_2.13-3.0.0 — ssh -i ~/Desktop/new-key-pair-emr.pem hadoop@ec2-44-195-77
[hadoop@ip-172-31-15-220 kafka_2.13-3.0.0]$ python3 put.py
Enter your ID: A20495530
Enter your name: Rishabh Jain
Enter your eye color: Brown
[hadoop@ip-172-31-15-220 kafka_2.13-3.0.0]$
```

Rishabh Jain
CSP-554
A20495530

On consumer Terminal

vim get.py (code file attached with the submission on blackboard).

```
from ensurepip import bootstrap
from kafka import KafkaConsumer
from json import loads

Consumer = KafkaConsumer('sample', bootstrap_servers = ['localhost:9092'],auto_offset_reset='earliest',enable_auto_commit=True,
group_id='my-group',value_deserializer = lambda x:loads(x.decode('utf-8')))

for i in Consumer:
    for key, value in i.value.items():
        print("key=%s value=%s" % (key,value))


Consumer.close()
```

Python3 get.py

```
[hadoop@ip-172-31-15-220 kafka_2.13-3.0.0]$ python3 get.py
key=MYID value=A20495530
key=MYNAME value=Rishabh Jain
key=MYEYECOLOR value=Brown
```