Project: Kafka + Spark Streaming + PySpark

CS570: Big-data Processing and Analytics

SFBU
Navya Kandimalla

Table Of Contents

Introduction

- Kafka is a distributed streaming platform that allows applications to publish and subscribe to streams of data.
- It is designed to be highly scalable and fault tolerant. Spark Streaming is an extension of the Apache Spark platform that enables real-time processing of streaming data.
- PySpark is the Python API for Apache Spark, which allows developers to write Python code to interact with the Spark platform.
- PySpark enables developers to create applications that can process streaming data in real-time, as well as batch data.

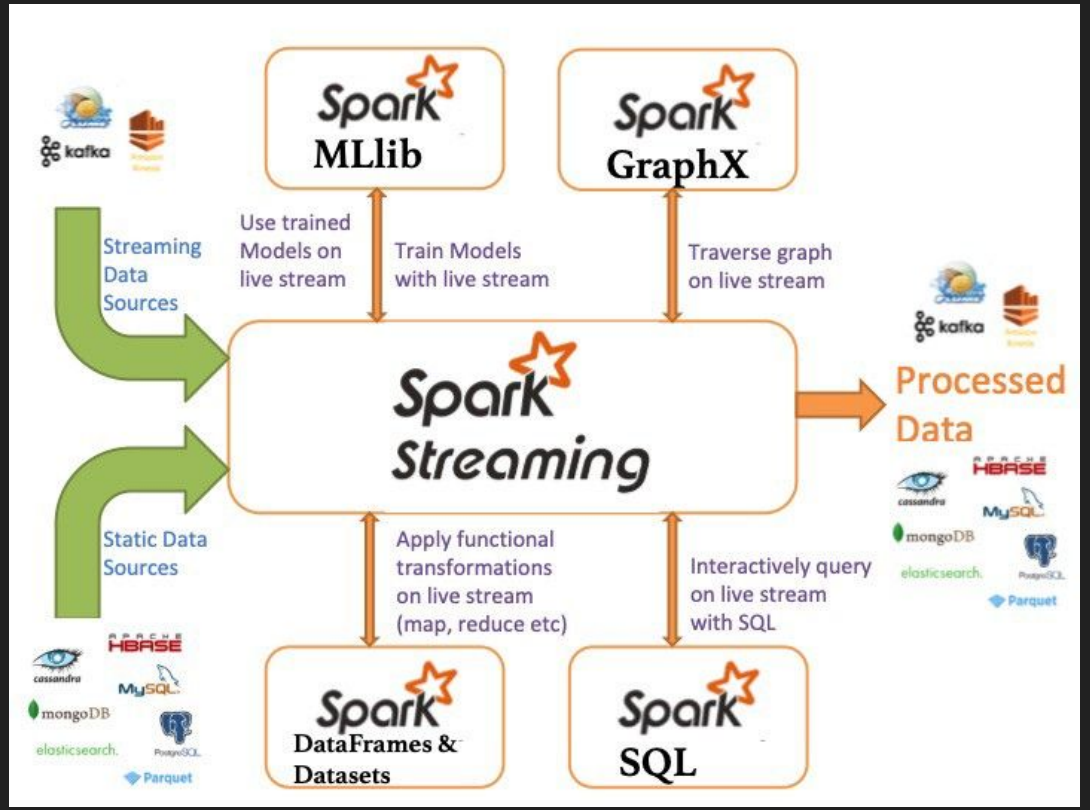
Design

- The design of a Kafka + Spark Streaming + PySpark system involves setting up a Kafka cluster to ingest streaming data, setting up a Spark cluster to process the data, and then using PySpark to write applications that can interact with the Spark cluster.

- The Kafka cluster can be configured to ingest data from various sources, such as web servers, databases, and message queues. The Spark cluster can then process the data in real-time, and the PySpark applications can be used to interact with

Design

The following diagram illustrates the design of a Kafka + Spark Streaming + PySpark system:



Implementation

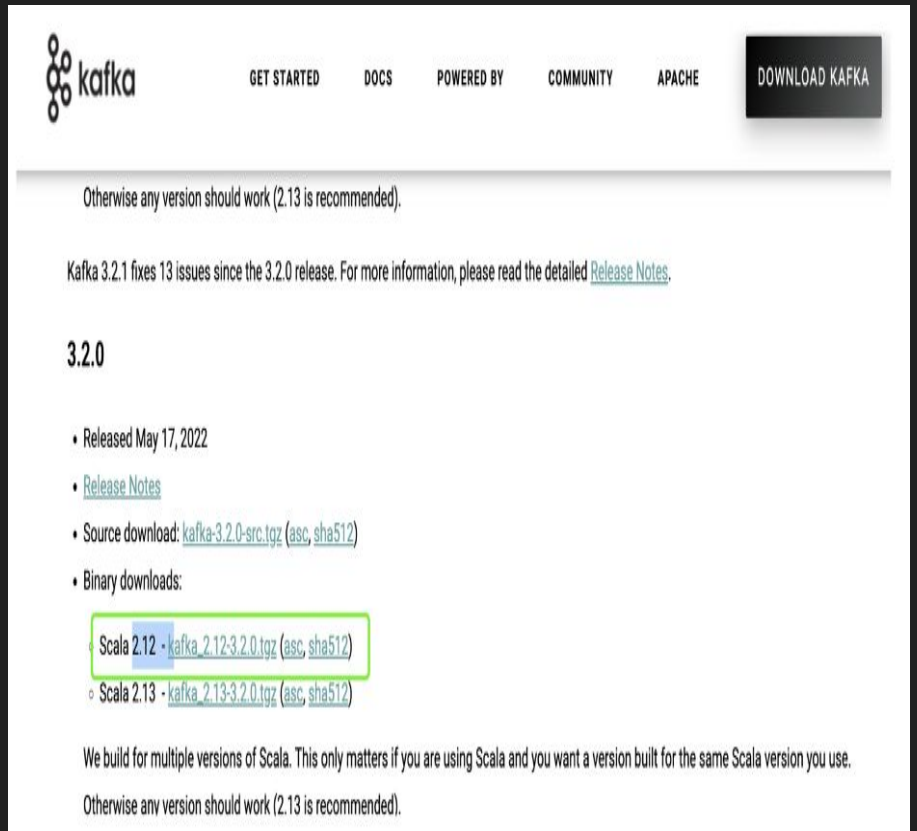
Kafka QuickStar—Apache Kafka + Kafka-Python

1.The latest version of Kafka binary distribution is available at

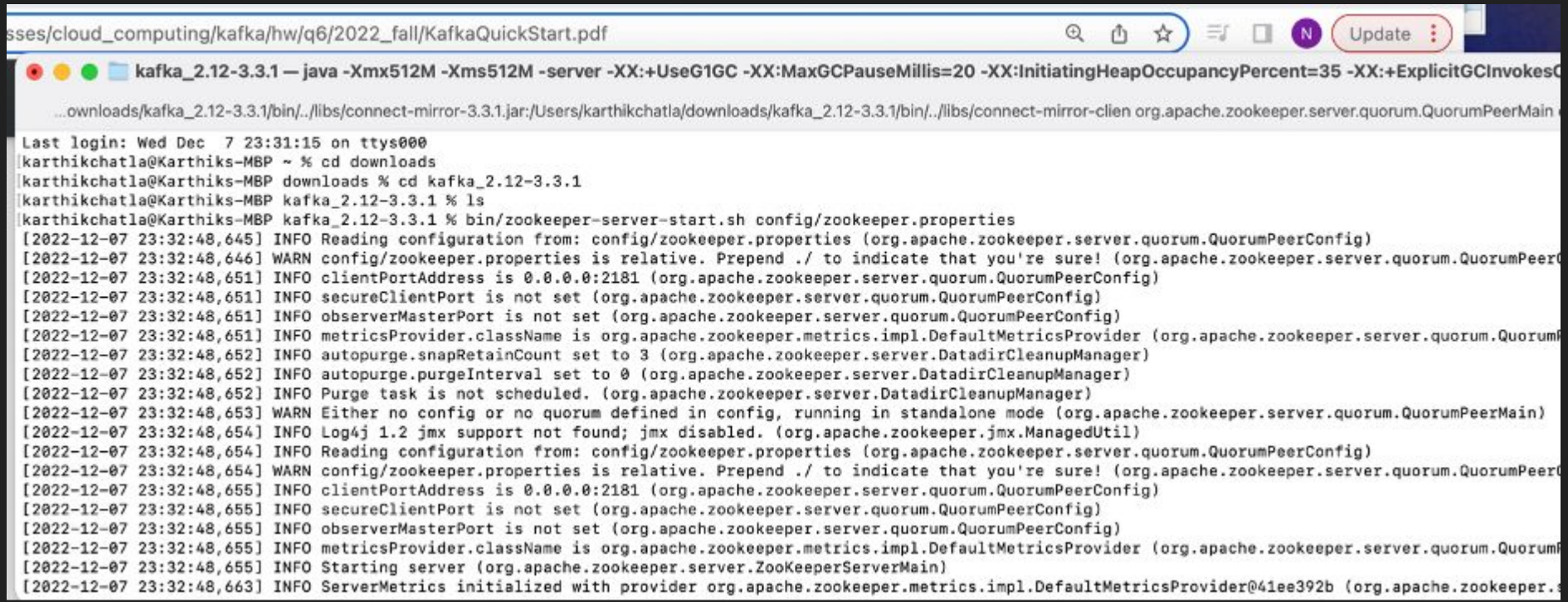
https://kafka.apache.org/downloads.

Test

2. Starting Zookeeper. Unzip it, get into the folders cd into it



3. Starting Kafka Brokers
Create another terminal, do not close zookeeper



4.Creating Kafka Topics.

Create another terminal, do not close zookeeper and kafka brokers

bin/kafka-topics.sh --create --topic input_recommend_product --zookeeper localhost:2181 --partitions 3 --replication-factor 1

```
[2022-12-09 00:27:51,172] INFO App info kafka.server for 0 unregistered (org.apache.kafka.common.utils.AppInfoParser)
[2022-12-09 00:27:51,173] INFO shut down completed (kafka.server.KafkaServer)
[2022-12-09 00:27:51,173] ERROR Exiting Kafka due to fatal exception during startup. (kafka.Kafka$)
kafka.zookeeper.ZooKeeperClientTimeoutException: Timed out waiting for connection while in state: CONNECTING
        at kafka.zookeeper.ZooKeeperClient.$anonfun$waitUntilConnected$3(ZooKeeperClient.scala:254)
        at kafka.zookeeper.ZooKeeperClient.waitUntilConnected(ZooKeeperClient.scala:250)
        at kafka.zookeeper.ZooKeeperClient.<init>(ZooKeeperClient.scala:108)
        at kafka.zk.KafkaZkClient$.apply(KafkaZkClient.scala:1980)
        at kafka.server.KafkaServer.initZkClient(KafkaServer.scala:503)
        at kafka.server.KafkaServer.startup(KafkaServer.scala:203)
        at kafka.Kafka$.main(Kafka.scala:109)
        at kafka.Kafka.main(Kafka.scala)
[2022-12-09 00:27:51,173] INFO shutting down (kafka.server.KafkaServer)
karthikchatla@Karthiks-MBP kafka_2.12-3.3.1 % bin/kafka-topics.sh --create --topic input_recommend_product --zookeeper localhost:2181 --partition
-replication-factor 1
Exception in thread "main" joptsimple.UnrecognizedOptionException: zookeeper is not a recognized option
        at joptsimple.OptionException.unrecognizedOption(OptionException.java:108)
        at joptsimple.OptionParser.handleLongOptionToken(OptionParser.java:510)
        at joptsimple.OptionParserState$2.handleArgument(OptionParserState.java:56)
        at joptsimple.OptionParser.parse(OptionParser.java:396)
        at kafka.admin.TopicCommand$TopicCommandOptions.<init>(TopicCommand.scala:567)
        at kafka.admin.TopicCommand$.main(TopicCommand.scala:47)
        at kafka.admin.TopicCommand.main(TopicCommand.scala)
karthikchatla@Karthiks-MBP kafka_2.12-3.3.1 %
```

5 Creating Producer and Consumer using Kafka-python

5.1 Create producer.py

```
from kafka import KafkaProducer
producer =
KafkaProducer(bootstrap_servers='localhost:9092')
producer.send('input_recommend_product', b'(1,
Main Menu), (2, Phone) ,(3, Smart Phone), (4,
iPhone)')
producer.close()
```

5.2 Create consumer.py

```
from kafka import KafkaConsumer
consumer = KafkaConsumer('input_recommend_product',
bootstrap_servers=['localhost:9092'])
for msg in consumer:
 print(msg)
```

5.3 Run comsumer.py first (you can run it in your IDE)

5.4 Create another terminal, run the producer.py

5.5 Go to the consumer terminal, you can see the result

Enhancement Ideas

- Kafka/Spark data pipelines are custom code based on some very stripped down system software, with all the dials available to you so that you can build exactly what you need.

- Building the system that matches the various impedances is not done automatically, but is something that has to be designed into your software.

Conclusion

- Kafka + Spark Streaming + PySpark is a powerful combination for real-time data processing.
- It allows for the ingestion of data from Kafka topics, the processing of data using Spark Streaming and PySpark, and the output of the processed data to other systems.
- This combination of technologies can be used to build powerful real-time data pipelines and applications.