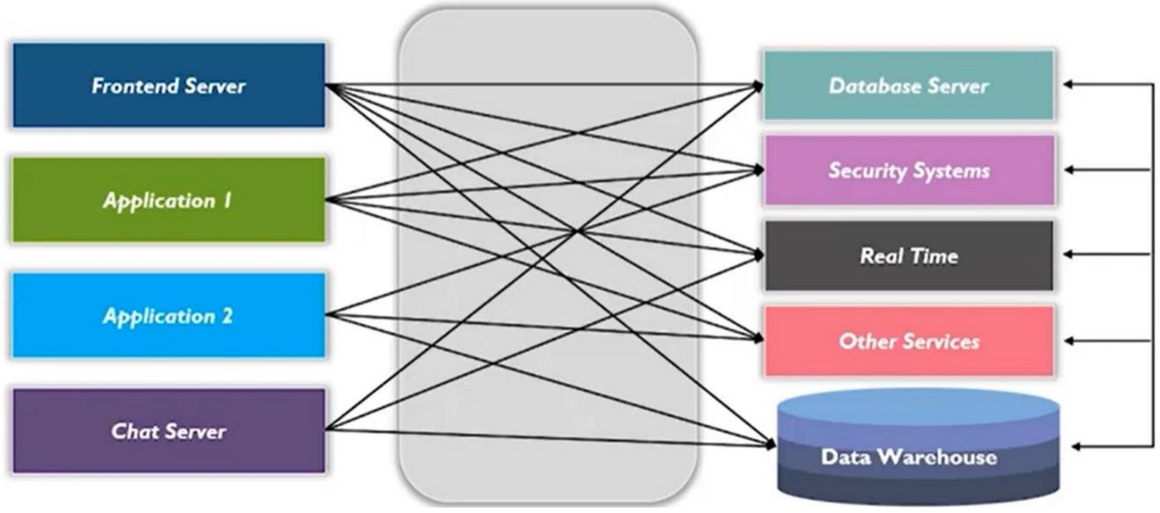
LSC Project: Kafka Integration with Spark

Magno Alessandro: 4478234

WHY KAFKA WAS NEEDED?

The current day industry is emerging with lots of real time data which needs to be processed in real time.

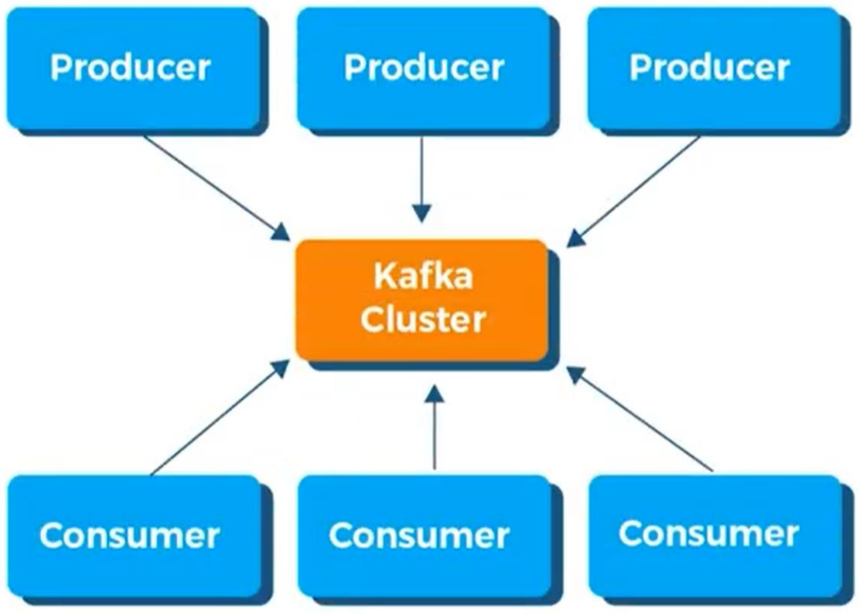
These days organizations have multiple servers at front end and back end like web or application server for hosting website or application. All of the servers will want to communicate with the database server and thus will have multiple data pipelines connecting all of them to the database server.



The data pipelines are getting more complex with the increase in number of systems:

- adding a new systems or server requires more data pipelines which will make the data flow complicated
- managing these data pipelines becomes very difficult as each data pipeline has its own set of requirements
- adding some pipelines or removing some pipelines is difficult in such case

HOW KAFKA SOLVES THESE PROBLEMS?



Kafka basically decouples the data pipelines so as you can see in the image, there is a cluster in the center (kafka cluster) and then we have consumers which can place a request and there are producers which send messages. So the producers send messages to the cluster from that the consumers can fetch them, in this way apache kafka reduces the complexity of data pipelines, makes simpler and manageable communication between systems. It is also easy to establish remote communication and send data across the network, so you can establish an asynchronous communication and send messages. Kafka also ensures that the communication is extremely reliable.

WHAT IS KAFKA?

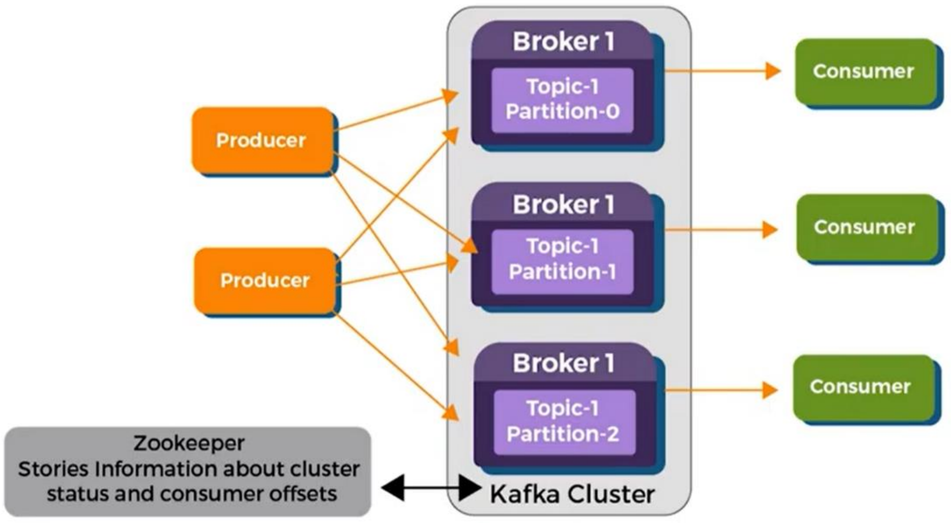
Apache kafka is an open source, distributed, publish-subscribe messaging system which manages and maintains the real time stream of data from different applications, websites, etc …

Kafka components:

- Broker: Kafka brokers are the servers that manage and mediate the conversation between 2 different systems. Brokers are responsible for the delivery of messages to the right party.
- Message: Messages are simply byte arrays and any object can be stored in any format by the developers. The format can be in String, JSON Avro and much more.

- Topic: In Kafka, all the messages are maintained in what we call topics. Messages are stored, published and organized in Kafka topics.
- Cluster: In Apache kafka, more than one brokers that is, a set of servers is collectively known as Kafka cluster. Also a group of computers, each having one instance of kafka broker.
- Producers: Producers are the processes that publish the data or messages to one or more topics. Producers are basically the source of data stream in kafka.
- Consumers: Consumers are the processes that read and process the data from topics by subscribing to one or more topics in the kafka cluster.
- Partitions: Every broker holds few partitions and each partition can be either a leader or a replica for a topic.

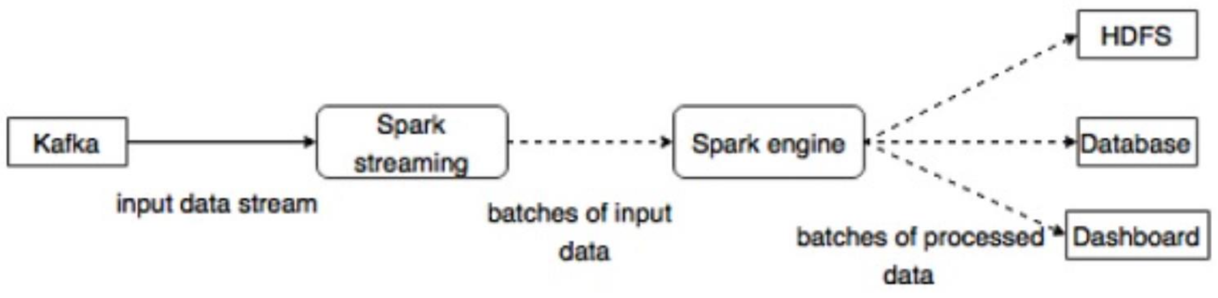
KAFKA ARCHITECTURE



The producers will send message to our topic at regular intervals, now brokers towards the messages in the partitions configured for that particular topic and have producer sends two messages and there exists two partitions, kafka will store one message in the first partition and the other message in the second partition. Consumer always subscribes to a specific topic, on receiving the message consumer sends an acknowledgement to the broker, on receiving the acknowledgement the offset is changed to the new value and is updated in the zookeeper.

INTEGRATION WITH SPARK

Kafka is a potential messaging and integration platform for Spark streaming. Kafka act as the central hub for real-time streams of data and are processed using complex algorithms in Spark Streaming. Once the data is processed, Spark Streaming could be publishing results into yet another Kafka topic or store in HDFS, databases or dashboards. The following diagram depicts the conceptual flow.



Exercise: words count through kafka

The idea is the following:

- download and install kafka here: https://kafka.apache.org/quickstart
- understand how kafka works
- run kafka and send messages
- there are two approaches for integrating spark with kafka: reciever-based and direct (no receivers), try both approaches with word count problem on the kafka stream you have just created

Questions

1. Explain the role of the offset
2. What is the role of the zookeeper in kafka?
3. Is it possible to use kafka without zookeeper?
4. What do you know about partition in kafka?
5. What are the types of traditional method of message transfer?
6. What are consumers or users?
7. What is the process for starting a kafka server?
8. Explain producer?
9. What is the difference between reciever-based and direct approaches for integrating spark with kafka?

Educational goals

Students will be able to understand and describe the architecture of kafka, exploring kafka producers and consumers for writing and reading messages, familiarizing with the commands and how to integrate kafka with spark streaming. Nowadays kafka is used by more than 2000 companies such as Linkedin, Airbnb, Netflix, Uber, Walmart and more companies are planning to use it in the future, so this lab activity is a first approach of this architecture whose study could be thorough in the future.

Solutions

Start Zookeeper

```
>bin\windows\zookeeper-server-start.bat config\zookeeper.properties
```

Start Kafka Broker

```
>bin\windows\kafka-server-start.bat config\server.properties
```

Create topic

```
>bin\windows\kafka-topics.bat --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic lab
```

Start Producer

```
>bin\windows\kafka-console-producer.bat --broker-list localhost:9092 --topic lab
```

#Send messages

>Hi how are you?

>Hi I am fine

Receive message

```
>bin\windows\kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic lab --from-beginning
```

Hi Luigi

Hi Alfredo

Code

```python
#version reciever-based approach
import sys
from pyspark import SparkContext
from pyspark.streaming import StreamingContext
from pyspark.streaming.kafka import KafkaUtils

if __name__ == "__main__":
    sc = SparkContext(appName="WordCountKafka")
    ssc = StreamingContext(sc, 2)
    broker, topic = sys.argv[1:]
    kvs = KafkaUtils.createStream(ssc, \
                                  broker, \
                                  "raw-event-streaming-consumer", \
                                  {topic:1})
    lines = kvs.map(lambda x: x[1])
    counts = lines.flatMap(lambda line: line.split(" ")) \
                  .map(lambda word: (word, 1)) \
                  .reduceByKey(lambda a, b: a+b)
    counts.pprint()
    ssc.start()
    ssc.awaitTermination()
```

```python

#version direct approach
import sys
from pyspark import SparkContext
from pyspark.streaming import StreamingContext
from pyspark.streaming.kafka import KafkaUtils

if __name__ == "__main__":
    sc = SparkContext(appName="WordCountKafka")
    ssc = StreamingContext(sc, 2)
    brokers, topic = sys.argv[1:]
    kvs = KafkaUtils.createDirectStream(ssc, [topic],{"metadata.broker.list": brokers})
    lines = kvs.map(lambda x: x[1])
    counts = lines.flatMap(lambda line: line.split(" ")) \
                  .map(lambda word: (word, 1)) \
                  .reduceByKey(lambda a, b: a+b)
    counts.pprint()
    ssc.start()
    ssc.awaitTermination()
```

Run

```
bin\spark-submit –packages org.apache.spark:spark-streaming-kafka-0-8_2.11:2.0.0 spark_kafka.py localhost:9092 lab
```

Output

```
— — — — — — — — — — — — — — — — — — — — — -
Time: 2021–01–19 13:04:10
— — — — — — — — — — — — — — — — — — — — — -
(u’Hi’, 2)
(u’Alfredo’, 1)
(u’Luigi’, 1)
```