Day5

Kafka

An Streaming Tool

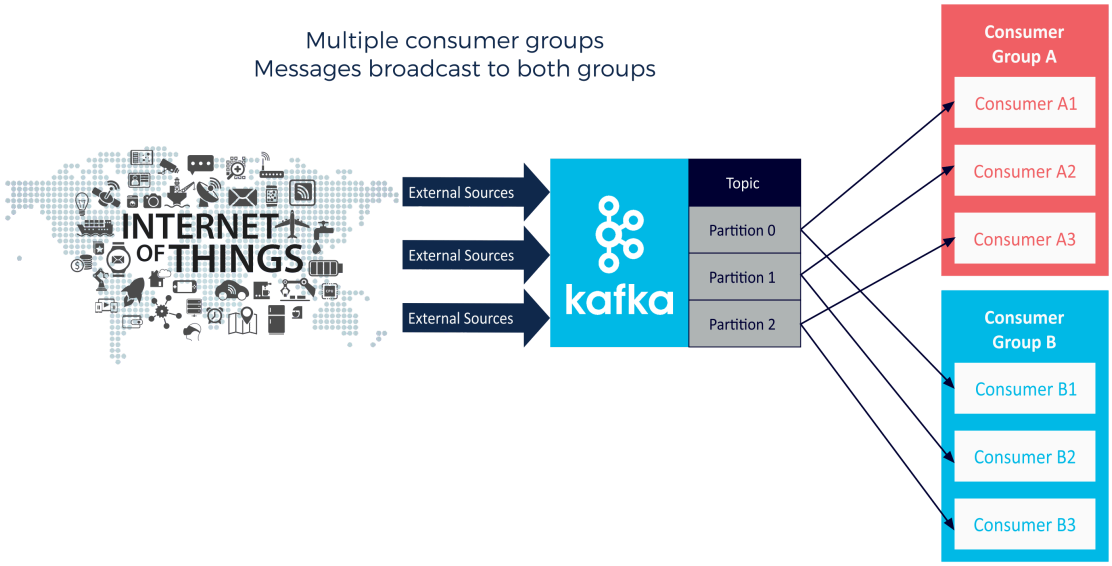
Source System, Target System, Integrator

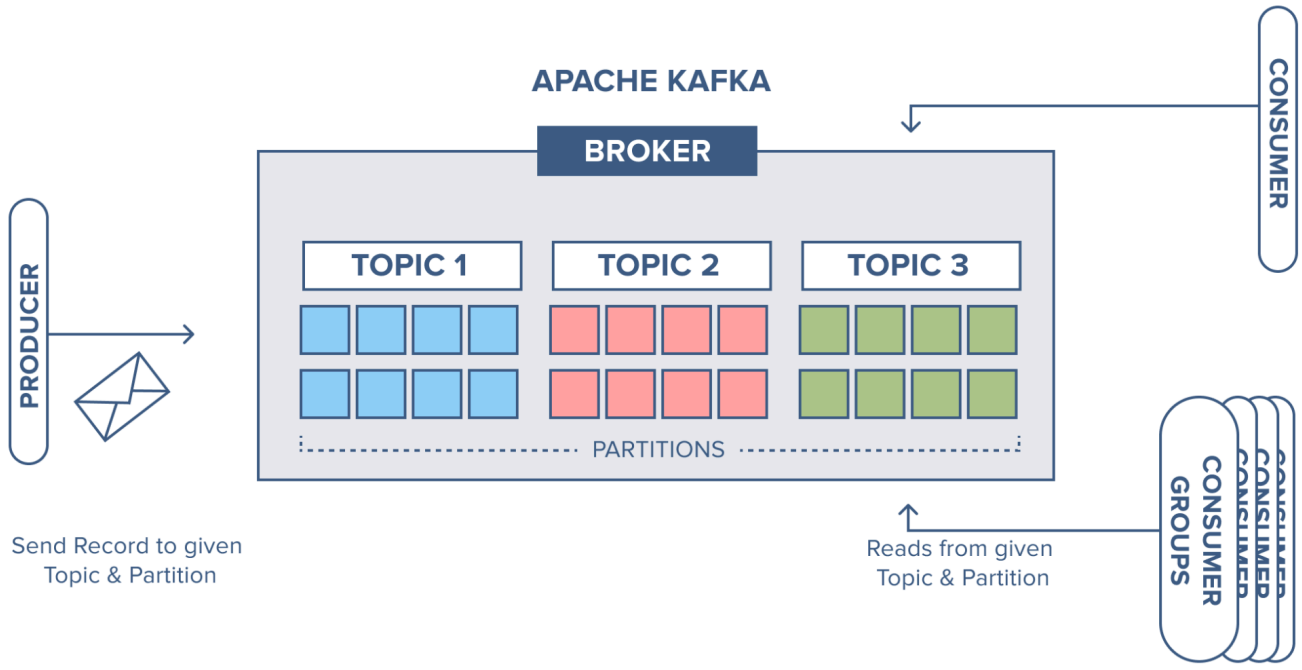
Integrator Challenges

- Data Protocol
- Data Format
- Data Schema and Evolution

decouple your data stream and your systems
Developed by LinkedIn

Streaming Data





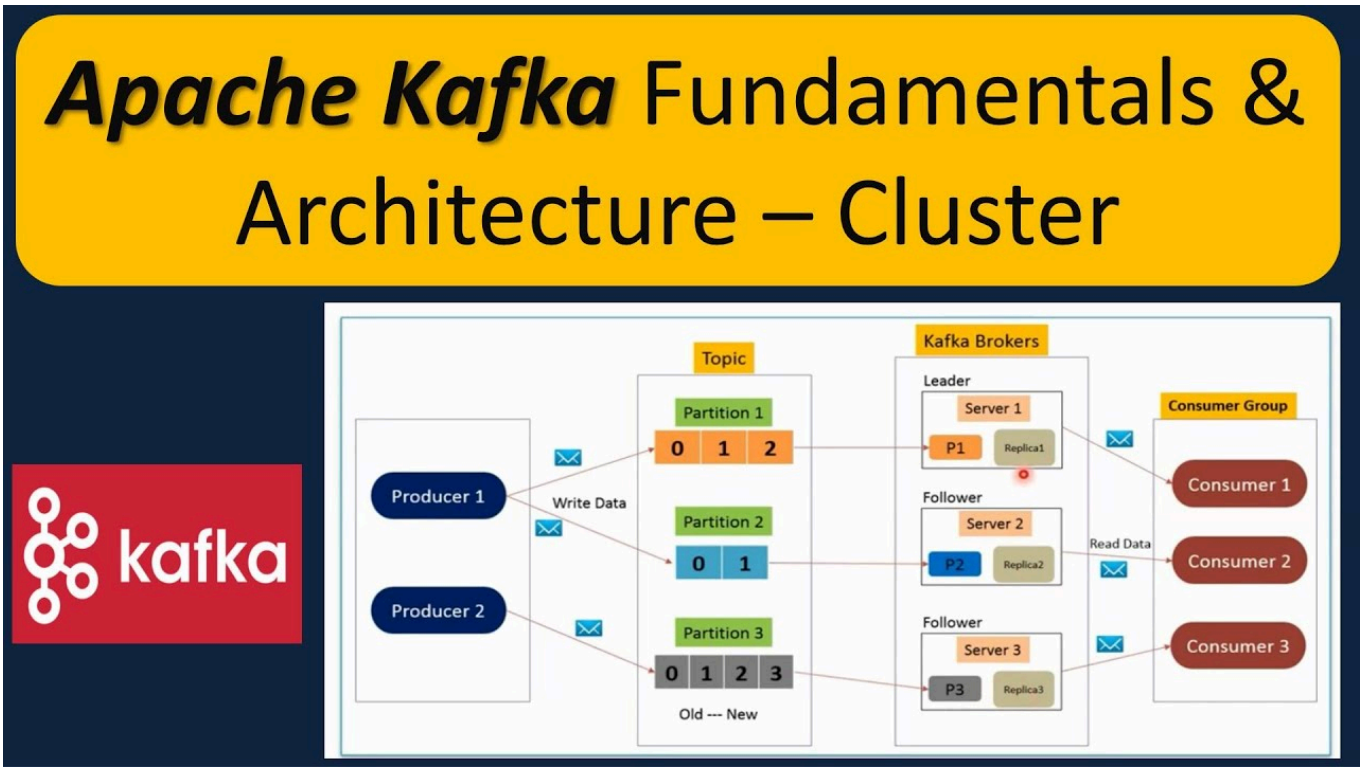
- distributed, resilient, architectures and fault tolerant
- scales horizontally
- low latency

Use Cases

- Messaging system

Kafka Cluster

Source Systems
Producers
Kafka Cluster -> Zookeeper
Consumers
Target Systems



Kafka as Producer
Spark as Consumer

Steps
Run Zookeeper
ls -ls

Server Start (Zookeeper + Kafka)

default port of Zookeeper: 2181
default port of Kafka: 9092

```
ls -lh

./run-kafka_zookeeper_server.sh -s start # Starts Zookeeper

jps -lm
#21622 sun.tools.jps.Jps -lm
#21224 org.apache.zookeeper.server.quorum.QuorumPeerMain
/home/talentum/kafka/config/zookeeper.properties

netstat -nputl | grep 21224
#(Not all processes could be identified, non-owned process info
# will not be shown, you would have to be root to see it all.)
tcp6       0      0 :::2181                 :::*                    LISTEN
21224/java
```

```
tcp6       0      0 :::36583                :::*                    LISTEN
21224/java

#
./run-kafka_server.sh -s start

jps
# 22017 Jps
# 21668 Kafka
# 21224 QuorumPeerMain

netstat -nputl | grep 21668
#(Not all processes could be identified, non-owned process info
# will not be shown, you would have to be root to see it all.)
tcp6       0      0 :::46777                :::*                    LISTEN
21668/java
tcp6       0      0 127.0.0.1:9092          :::*                    LISTEN
21668/java

./run-kafka_server.sh -s stop
./run-kafka_zookeeper_server.sh -s stop # Stop Zookeeper
```

jps - java process stack
ps

Kafka Topics

```
kafka-topics.sh --zookeeper 127.0.0.1:2181 --topic first_topic --create

kafka-topics.sh --zookeeper 127.0.0.1:2181 --topic first_topic --create --
partitions 3

kafka-topics.sh --zookeeper 127.0.0.1:2181 --topic first_topic --create --
partitions 3 --replication-factor 1
# WARNING: Due to limitations in metric names, topics with a period ('.') or
underscore ('_') could collide. To avoid issues it is best to use either, but
not both.
# Created topic first_topic.

kafka-topics.sh --zookeeper 127.0.0.1:2181 --list
# first_topic

kafka-topics.sh --zookeeper 127.0.0.1:2181 --topic first_topic --describe
# Topic:first_topic PartitionCount:3    ReplicationFactor:1 Configs:
```

```
#   Topic: first_topic  Partition: 0    Leader: 0   Replicas: 0 Isr: 0
#   Topic: first_topic  Partition: 1    Leader: 0   Replicas: 0 Isr: 0
#   Topic: first_topic  Partition: 2    Leader: 0   Replicas: 0 Isr: 0

kafka-topics.sh --zookeeper 127.0.0.1:2181 --topic second_topic --delete
#Topic second_topic is marked for deletion.
#Note: This will have no impact if delete.topic.enable is not set to true.
```

each kafka broker has one replica

Classworks

Q. Create a new topic, having name as second_topic, having six partitions, and then the replication factor, which is going to be one.
List all the topics to confirm the creation of the above topic.

```
kafka-topics.sh --zookeeper 127.0.0.1:2181 --topic second_topic --create --
partitions 6 --replication-factor 1

kafka-topics.sh --zookeeper 127.0.0.1:2181 --list

kafka-topics.sh --zookeeper 127.0.0.1:2181 --topic second_topic --delete
```

Kafka Producer

```
./kafka-console-producer.sh
# Error

kafka-console-producer.sh --broker-list 127.0.0.1:9092 --topic first_topic
>Message1
>Message2
>Message3
# Standard Input

kafka-console-producer.sh --broker-list 127.0.0.1:9092 --topic first_topic --
producer-property acks=all
```

Working in asynchronous Form

Kafka Consumer

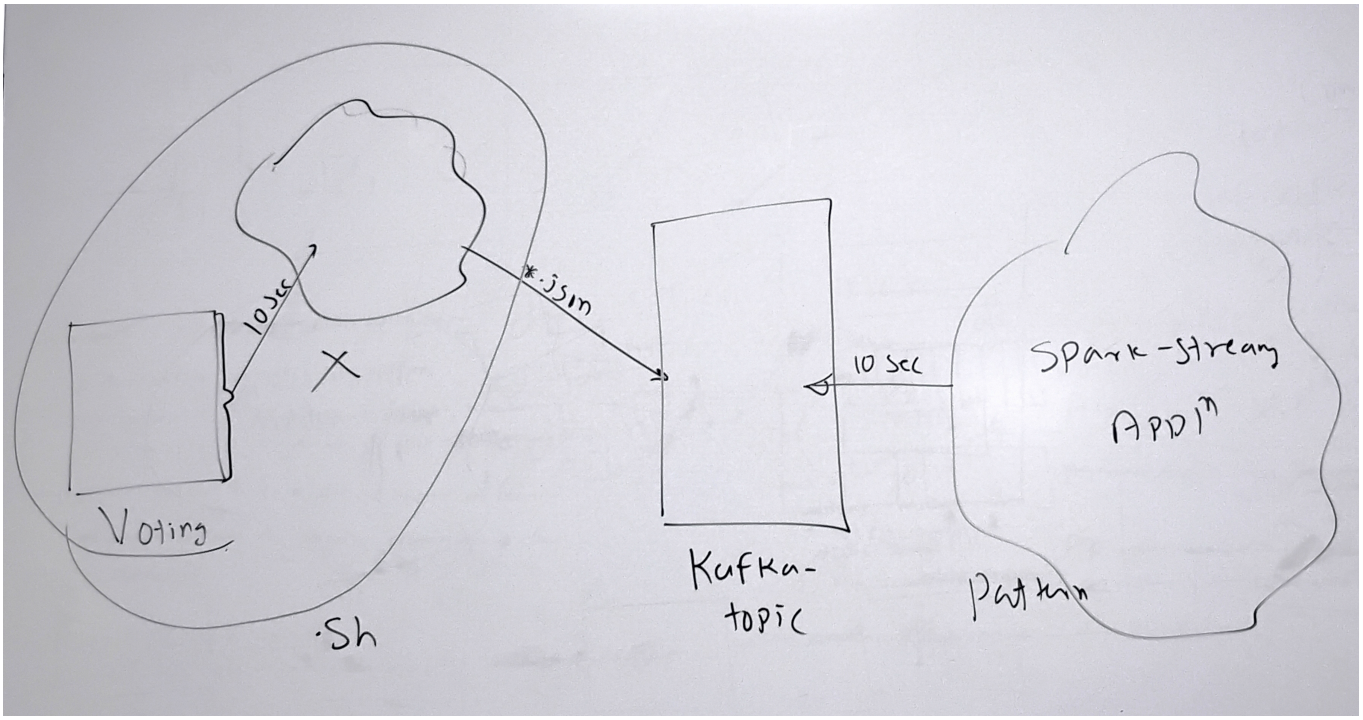
```
kafka-console-consumer.sh --bootstrap-server 127.0.0.1:9092 --topic
first_topic
# Consumes the message in real time. Both should be running

kafka-console-consumer.sh --bootstrap-server 127.0.0.1:9092 --topic
first_topic --from-beginning
# Print all messages (Doesnot Maintains order)

kafka-console-consumer --bootstrap-server 127.0.0.1:9092 --topic first_topic -
-group my-first-application
```

Message Consumption in Round Robin Fashion to consumers in same group
If number of consumer is equal to partitions (Round Robin Consumptions)
If number of consumer is greater to partitions (Extra are in inactive state)
If case of crash (Extra consumer gets an active state)

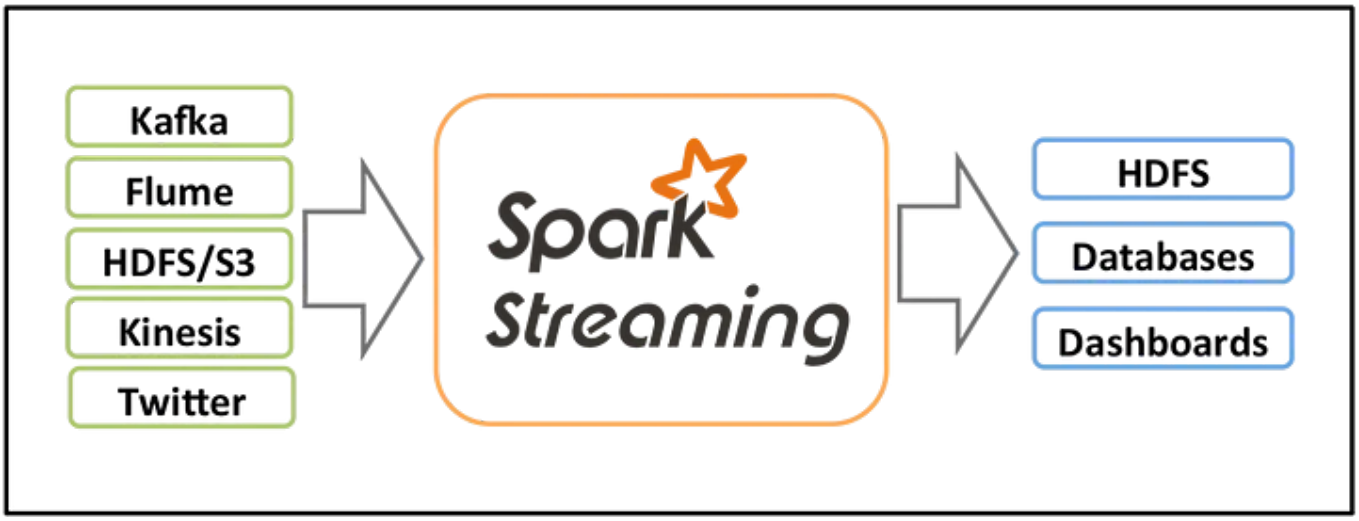
Classwork



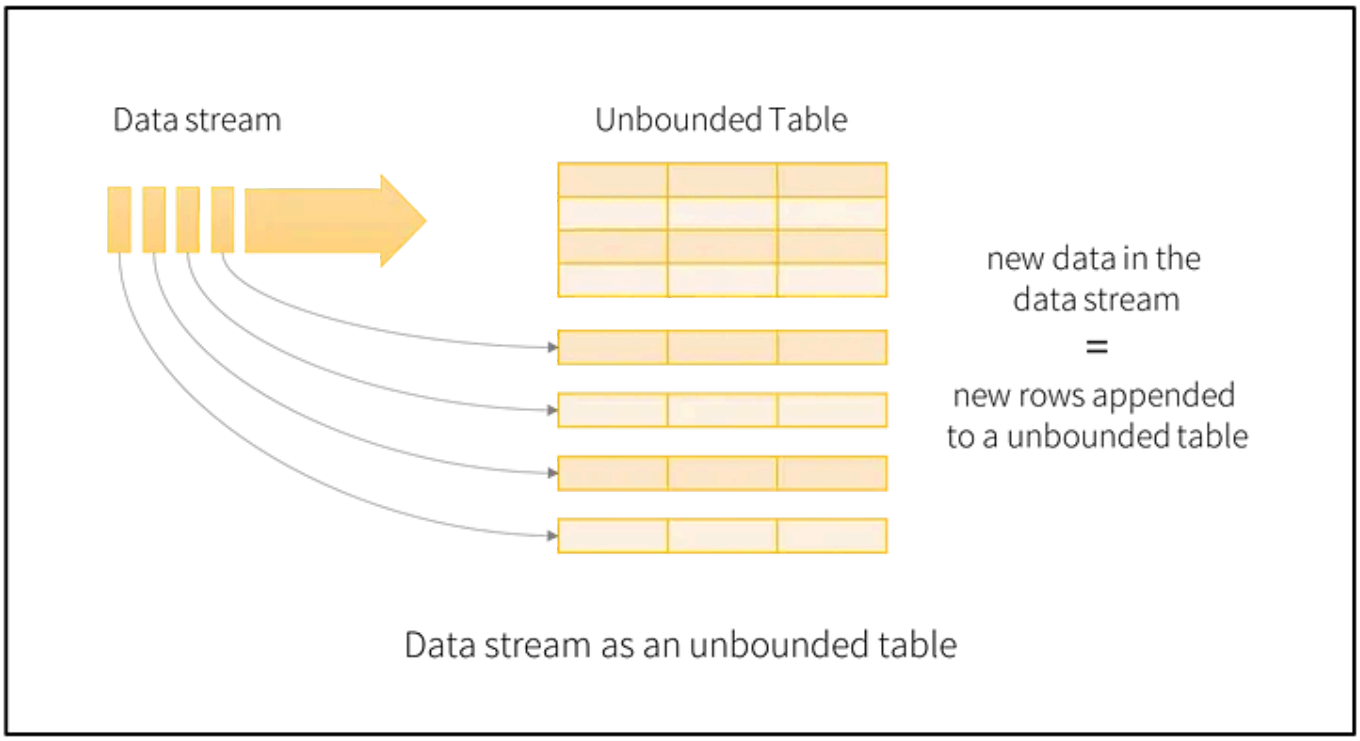
create sh script to produce messages continuously (json format)

microbatch interval

Spark Streaming



Input sources generate data like Kafka, Flume, HDFS/S3, etc. Spark Streaming engine processes incoming data from various input sources. Sinks store processed data from Spark Streaming engine like HDFS, relational databases, or NoSQL datastores.



Spark will process data in micro-batches which can be defined by triggers.

https://medium.com/expedia-group-tech/apache-spark-structured-streaming-first-streaming-example-1-of-6-e8f3219748ef

Output Modes

Append Mode, Update Mode, Complete Mode

Classwork

First Spark Streaming example using rate **source** and console **sink**. Rate source will auto-generate data which we will then print onto a console.

Use ReadMe from `1_Spark-S-Streaming_Demos`
`1_Spark-S-Streaming_Demos` contains all `.py` files

```
mkdir -p ~/Spark-S-Streaming_Demos/
cp -r ~/shared_folder/1_Spark-S-Streaming_Demos/* ~/Spark-S-Streaming_Demos/
./Start-Hadoop-Hive.sh #Start Hadoop

# demo1.py
uncommment spark-submit demo1.py at run-demo.sh
./run-demo.sh

# demo2.py
# input source test
uncommment spark-submit demo2.py at run-demo.sh
nc -lk 9999 # terminal1
./run-demo.sh # terminal2
# Give Sentence at socket -> terminal1
# Output at terminal2

# demo3.py
# input source File
# /home/talentum/Spark-S-Streaming_Demos/data/stream/ # stream sink
uncommment spark-submit demo3.py at run-demo.sh
./run-demo.sh
cp -r /home/talentum/Spark-S-Streaming_Demos/data/stocks/MSFT_2017.csv
/home/talentum/Spark-S-Streaming_Demos/data/stream/
cp -r /home/talentum/Spark-S-Streaming_Demos/data/stocks/GE_2017.csv
/home/talentum/Spark-S-Streaming_Demos/data/stream/
cp -r /home/talentum/Spark-S-Streaming_Demos/data/stocks/MSFT_2016.csv
/home/talentum/Spark-S-Streaming_Demos/data/stream/
cp -r /home/talentum/Spark-S-Streaming_Demos/data/stocks/AMZN_2017.csv
/home/talentum/Spark-S-Streaming_Demos/data/stream/

# demo4.py
# input source File
# /home/talentum/Spark-S-Streaming_Demos/data/stream/ # stream sink
```