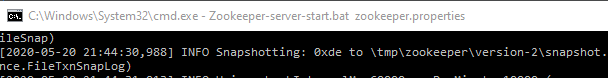
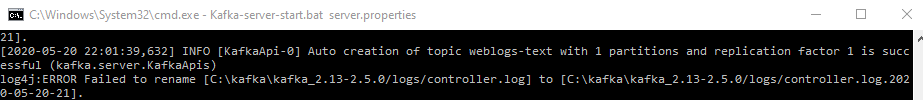
**KAFKA STREAMS**

1. Specify the necessary libraries in build.gradle file

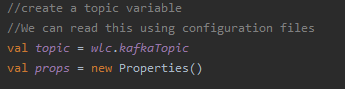
compile group: 'org.apache.kafka', name: 'kafka-clients', version: '0.8.2.1'  
compile 'org.apache.kafka:kafka\_2.11:0.8.2.1'

1. Import the necessary libraries
2. import org.apache.kafka.clients.producer.{KafkaProducer, Producer, ProducerConfig, ProducerRecord}
3. Create new log producer class
   1. It would have a single instance which would run on localhost in the vm which we have up and running or run zookeeper kafka brokers locally as shown below

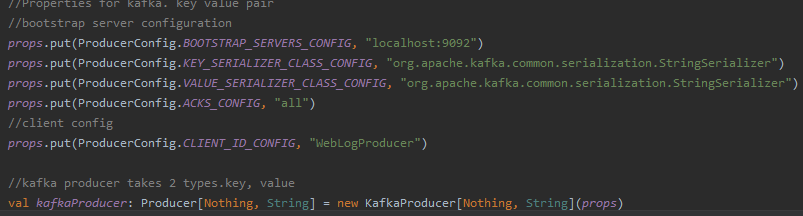




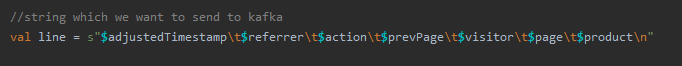
* 1. Specify which topic we want to send data to

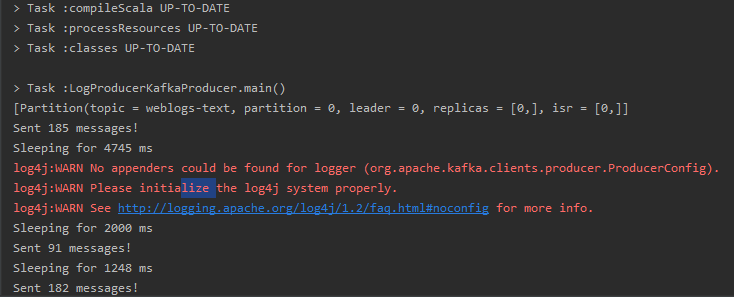


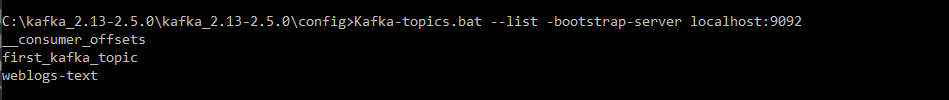
* 1. Create a new instance of kafka producer .It would have set of properties including brokers address,key serializers, acks\_config and client\_id\_config



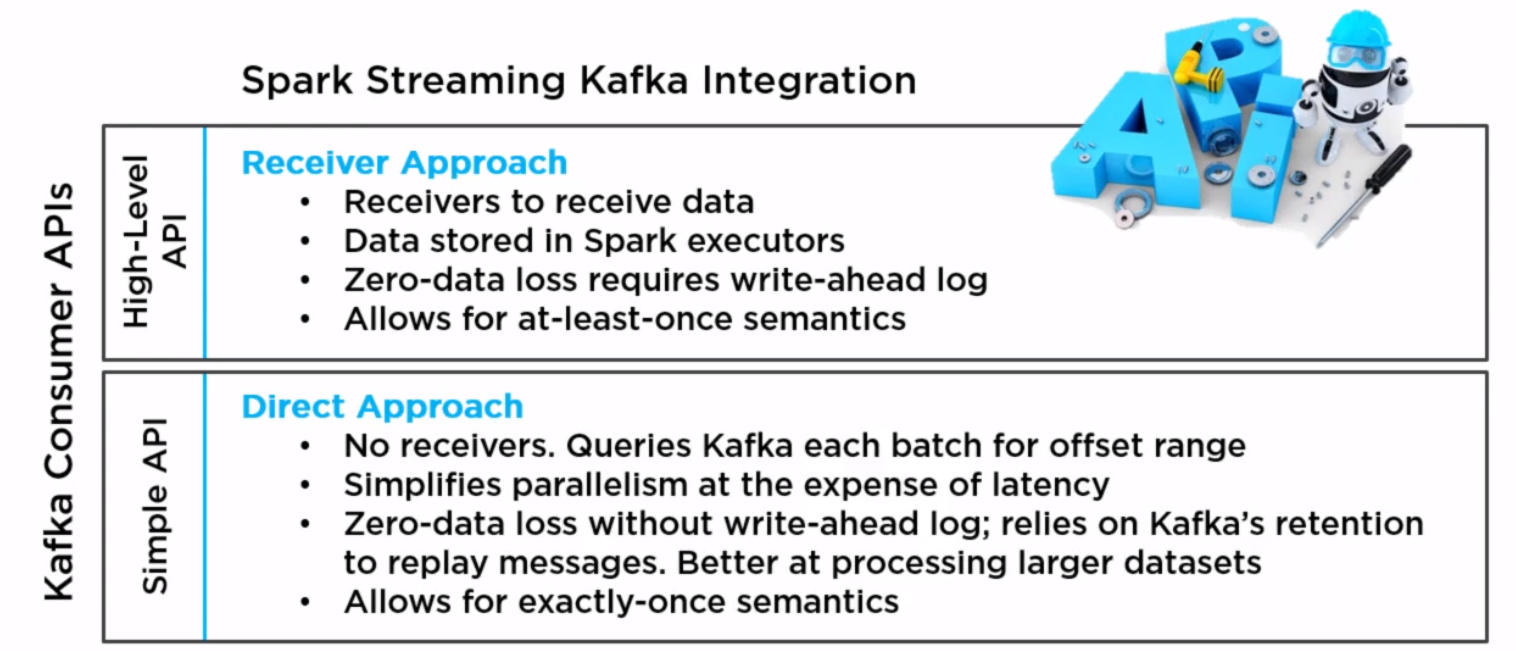
* 1. New instance of sting would be sent to kafka



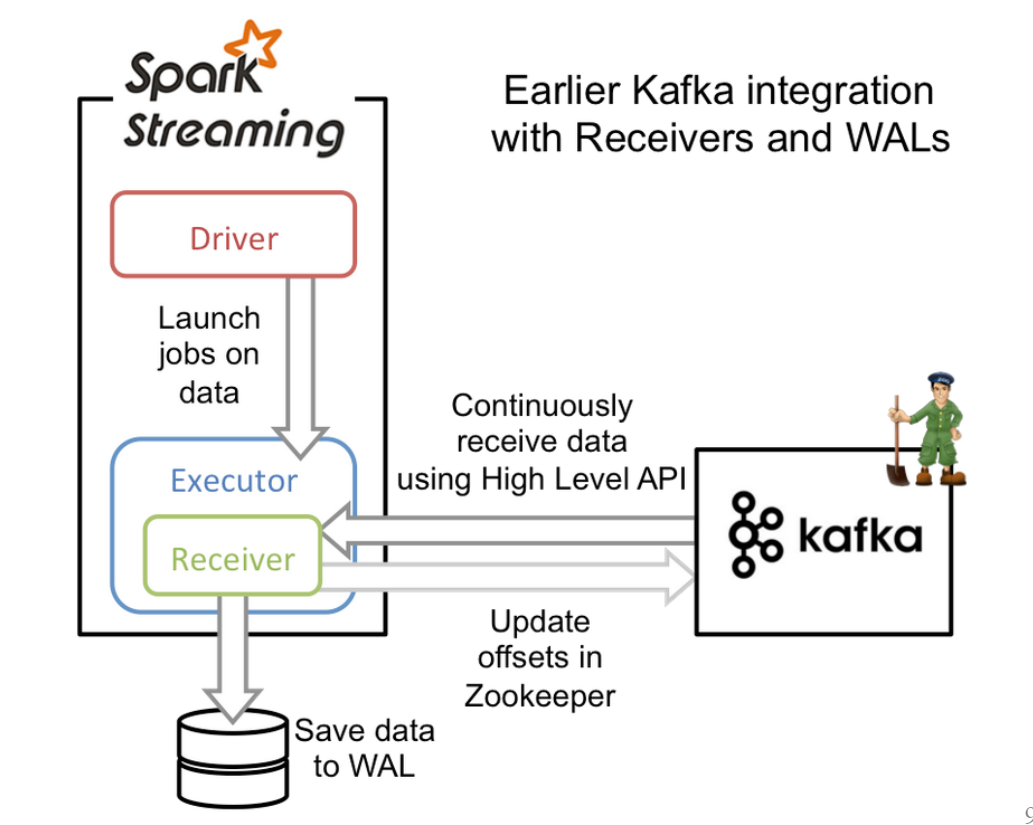
1. Run the kafklogproducer
2. After running the kafka producer we can see that kafka producer has been created.



**KAFKA INTEGRATION**



**KAFKA INTEGRATION WITH RECIEVERS**



**Pro:**

* WAL design could work with non-Kafka data sources

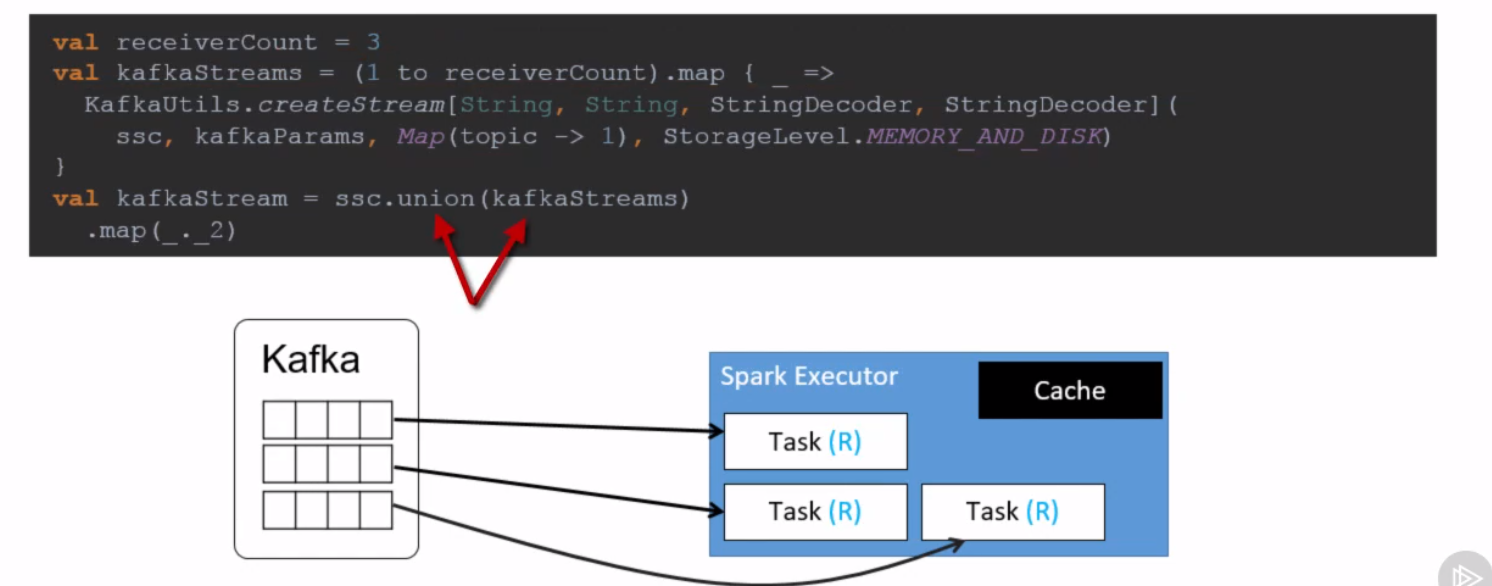
**Con:**

* Long running receivers make parallelism awkward and costly
* Duplication of write operations
* Dependent on HDFS
* Must use idempotence for exactly-once
* No access to offsets, can't use transactional approach

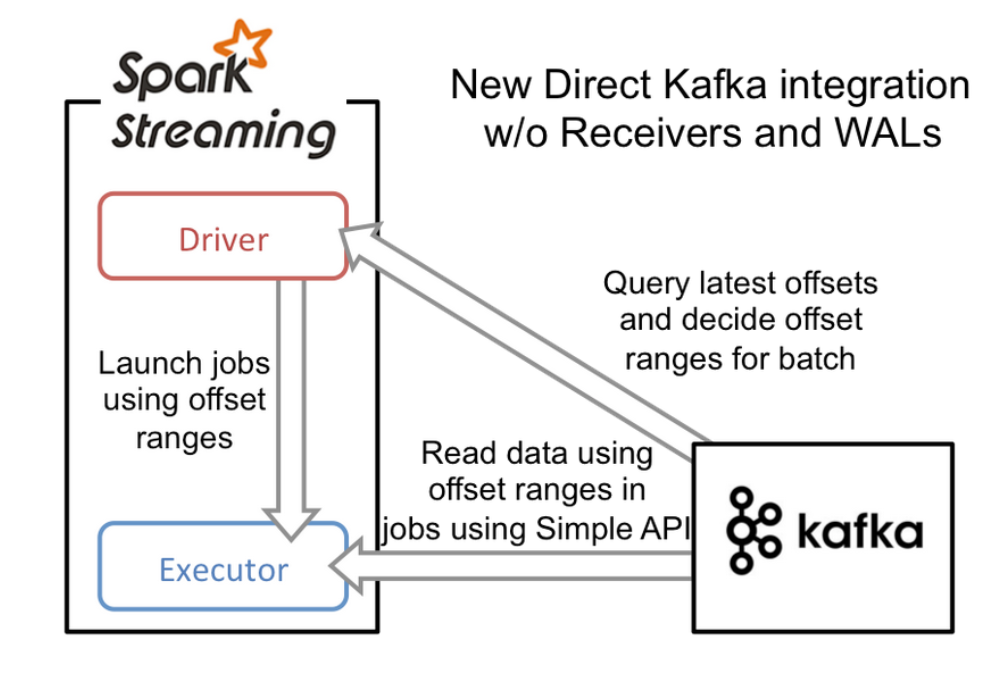
Note:

1)We can increase the level of parallelism by increasing the no of receivers. We do that by using range collections, map over every item and create a stream

2) result would be sequence of RecieverInput stream



**KAFKA INTEGRATION WITHOUT USING RECIEVERS**



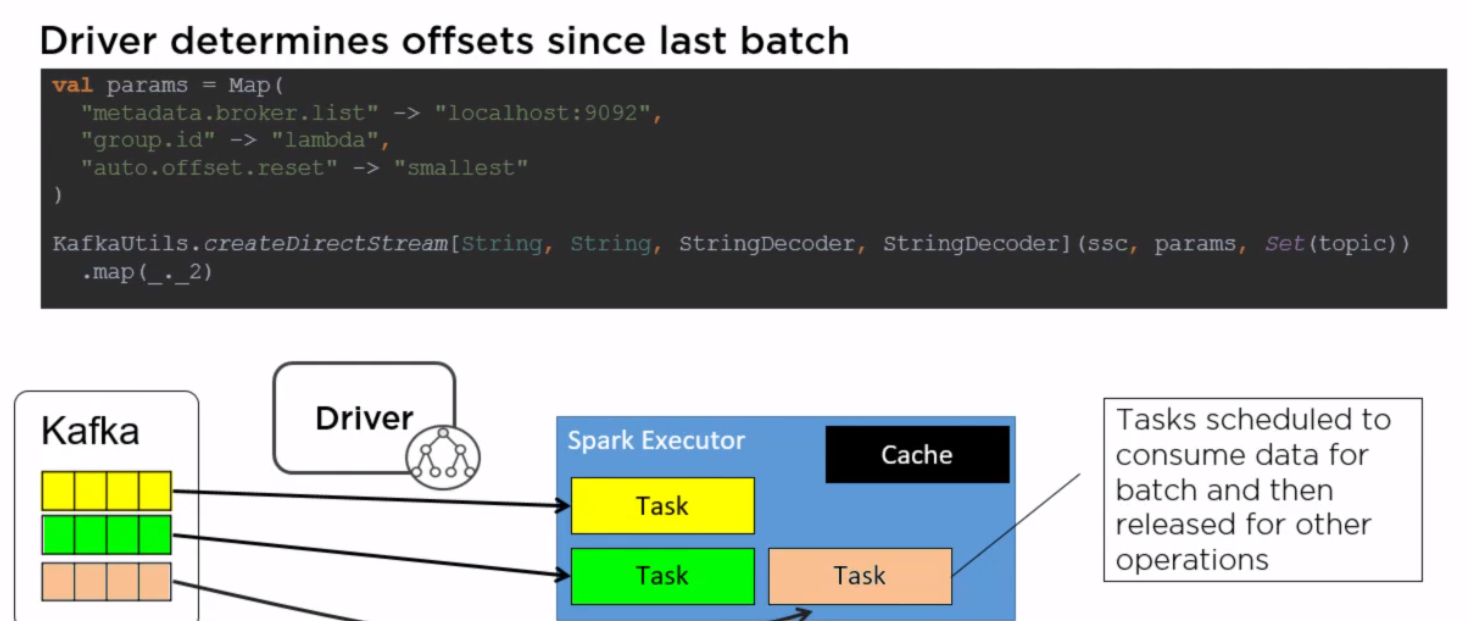
* Relies on spark checkpoint to save the offsets
* Does not rely on zookeeper

**Pro:**

* Spark partition 1:1 Kafka topic/partition, easy cheap parallelism
* No duplicate writes
* No dependency on HDFS
* Access to offsets, can use idempotent or transactional

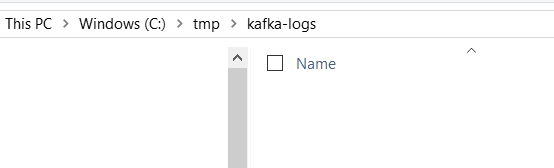
**Con:**

* Specific to Kafka
* Need adequate Kafka retention (OffsetOutOfRange is your fault)
* Do not get option to control the parallelism. If kafka topic has 20 partitions (Not a high-volume topic), the driver ends up having to schedule and eventually consume 20 tasks on the executors



**KAFKA LOGS**

Kafka stores all messages in logs in their respective nodes at the location specified in log.dir. We should remove all these messages for a topic from all nodes



Note: we can manually delete files from this folder

**KAFKA RETENTION HOURS**

* Policy can bet set to delete segments after a period, or after given size has accumulated.
* A segment would be deleted whenever \*either\* of these criteria are met.
* Deletion always happens from the end of the log.

Note: These settings can be configured in server.properties file

