

Kafka and Streaming

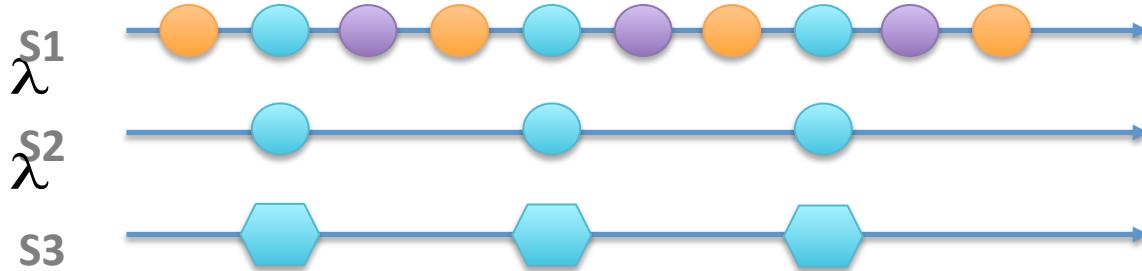
Introduction

- Kafka Streaming: An alternative for streaming to and from Kafka
 - Part of the Apache Kafka
 - Powerful
 - Highly scalable, fault-tolerant
 - Rich feature set with support for both stateless and stateful processing
 - Support for fault-tolerant local state
 - Lightweight
 - Just a library integrated in Kafka (no external dependencies)
 - No need for dedicated clusters
 - Near real-time
 - Millisecond processing latency
- Kafka and Spark / Flink

Two API's

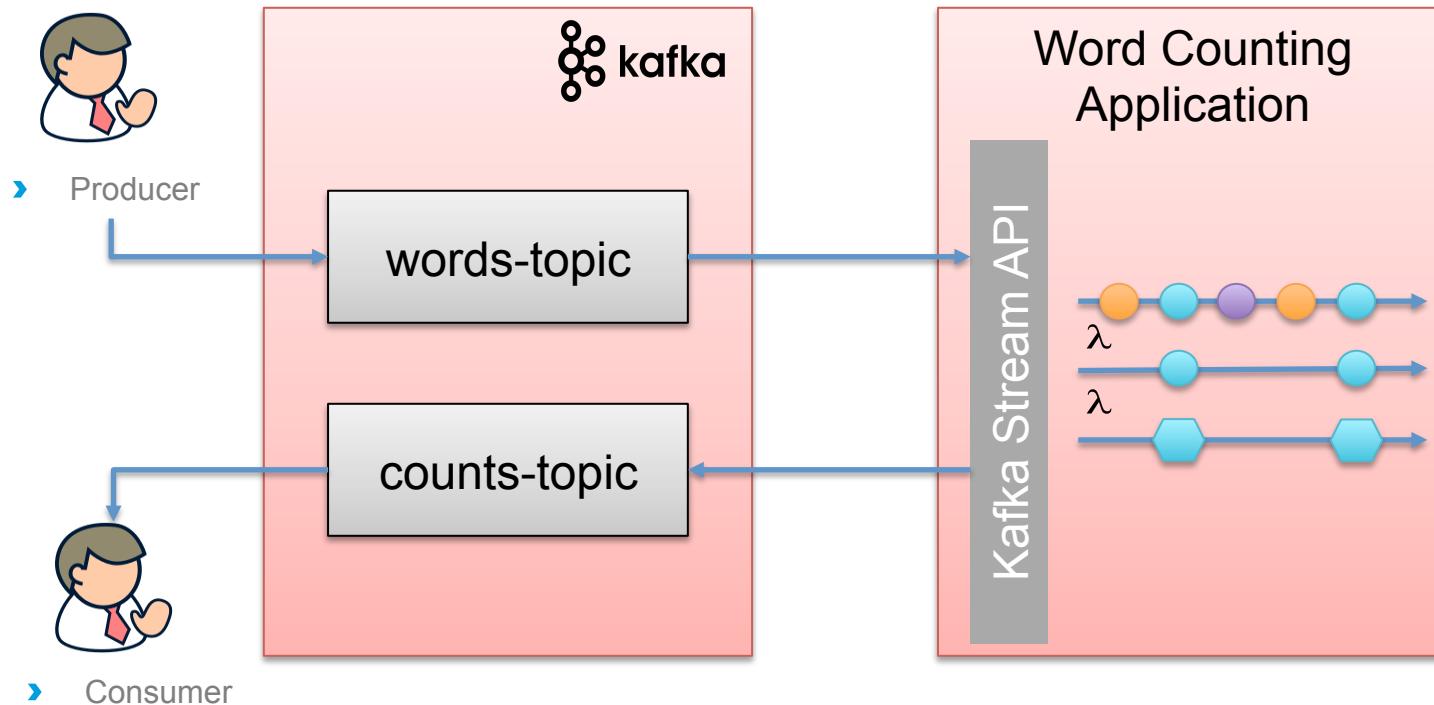
- Kafka supports a low level API that can be used to manipulate streams
- We'll focus on the Kafka Stream `DSL`
 - Higher level
 - Closer to other streaming ideas
 - Brining in some of the functional programming paradigms

What is Stream Processing?



- Programming with streams of data (even unbounded streams)
- Typical traits
 - Functional
 - Apply a dataflow or a sequence of functions to the streams
 - Combining functions provides a powerful expression language (lambda expression)
 - Reactive
 - Process incoming stream data immediately upon receipt

The Obligatory Word Count Example



Kafka Stream Classes

- StreamsConfig
 - A wrapper around a map used for configuration
- KStreamBuilder
 - The key class for setting up the stream procoessing
 - Based on the GoF Builder Pattern
 - Fluent API
 - Constructs the processing dataflow
- KafkaStreams
 - A wrapper of the actual stream

Enable Kafka Stream (Using Maven)

```
<dependency>
    <groupId>org.apache.kafka</groupId>
    <artifactId>kafka-streams</artifactId>
    <version>0.10.1.1</version>
</dependency>
```

- Kafka Streams are distributed as a simple jar file
- Simply include the jar file in your program and you'll be able to use the streaming library

Programming to Stream

```
// Configure
Properties props = new Properties();
props.put(StreamsConfig.CLIENT_ID_CONFIG, "...")
...
StreamsConfig config = new StreamsConfig(props);

// Serialization/deserialization
Serde<String> serDeser = Serdes.String();

// Create the data processing pipeline
KStreamBuilder bld = new KStreamBuilder();
bld.stream(serDeser, serDeser, "topic")
.map(...)

// Create stream
KafkaStreams sp = new KafkaStreams(bld, config);

// Start stream
kafkaStreams.start();
```

Word Count Streaming

```
StreamsConfig config = new StreamsConfig(props);
Serde<String> serde = Serdes.String();

KStreamBuilder bld = new KStreamBuilder();
bld.stream(serde, serde, "words-topic")
    .flatMapValues(text -> asList(text.split(" ")))
    .map((key, word) -> new KeyValue<>(word, word))
    .countByKey(serde, "Counts")
    .toStream()
    .map((word, count) ->
        new KeyValue<>(word, word + ":" + count))
    .to(serde, serde, "counts-topic");

KafkaStreams s= new KafkaStreams(bld, config);
s.start();
```

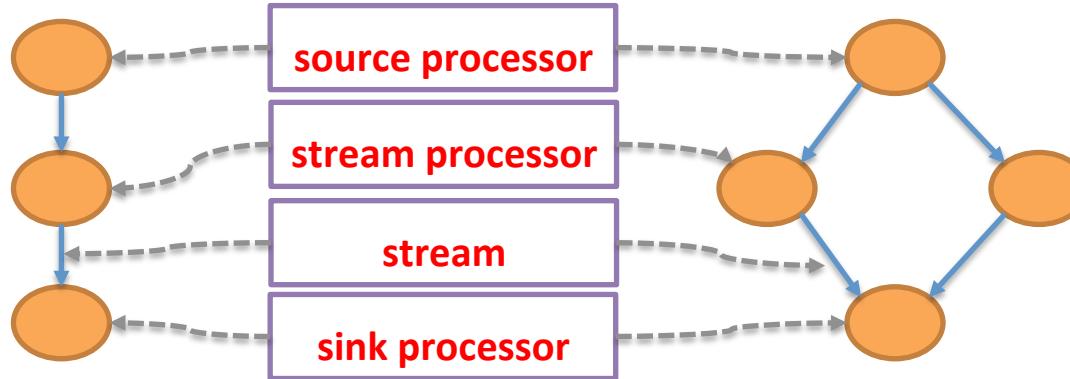
kafkacat – A Useful Tool

- kafkacat is a tool created by Magnus Edenhill
 - <https://github.com/edenhill/kafkacat>
 - Doesn't require JVM
 - Similar to 'netcat'
 - Can act as a Producer and consumer
 - Useful also to query metadata from Kafka
- To provide the word list to the application, we could simply use kafkacat instead of writing new producers and consumers
- Use as a producer
 - kafkacat –P –b myBroker –t topic1
- Use as a consumer
 - kafkacat –b myBroker –G mygroup topic1 topic 2

Installing kafkacat

- The installation of kafkacat is trivial on Mac most Linux distributions
 - Mac
 - brew install kafkacat
 - Ubuntu or Debian
 - [sudo] apt-get install kafkacat
- Our Kafka docker image is using Alpine distribution so if you want to run it in the docker image you'll have to:
 - apk add --update alpine-sdk bash python cmake
 - curl https://codeload.github.com/edenhill/kafkacat/tar.gz/master | tar xzf - && cd kafkacat-* && bash ./bootstrap.sh

Processor Topology



- The topology defines the computational logic of the data processing pipeline
- The topology typically forms a chain, but more advanced computation may be defined as a graph

Processing and Time

- In streaming time is an important design consideration
 - When working with concepts such as windowing, it is essential to establish which time to use
- Event-time
 - The point in time when an event or data record was created by the source
- Processing-time
 - The point in time when an event was processed by the stream processor
- Ingestion-time
 - The point in time when an event was stored in a topic partition by one of the Kafka brokers

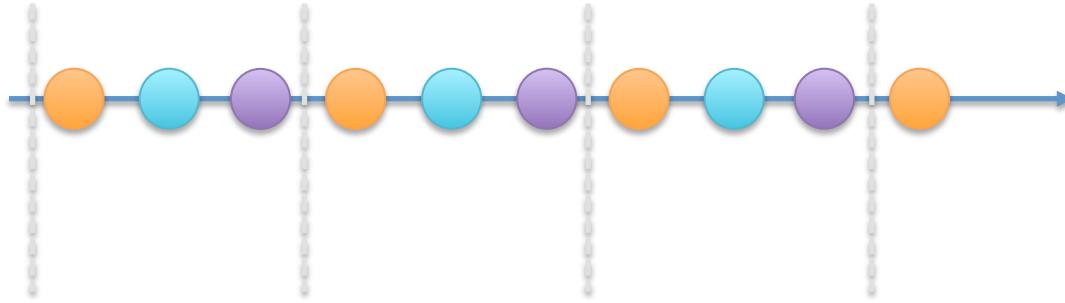
Stateful Stream Processing

- Typically we try to make our stream processing stateless
- However, for some use cases, stateful streaming may be essential
- The Kafka Stream DSL provides support for stateful processing

Stream vs. Table

- Kafka streams introduces the concept of tables, where
 - A stream can be viewed as a table: KStream → KTable
 - A table can be viewed as a stream: KTable → KStream
 - (called stream-table duality)
- Streams as tables
 - A stream can be considered a changelog of a table
 - One can convert the stream into a table by replaying the event stream
- Table as streams
 - A table can be converted into a stream by iterating over each key-value entry

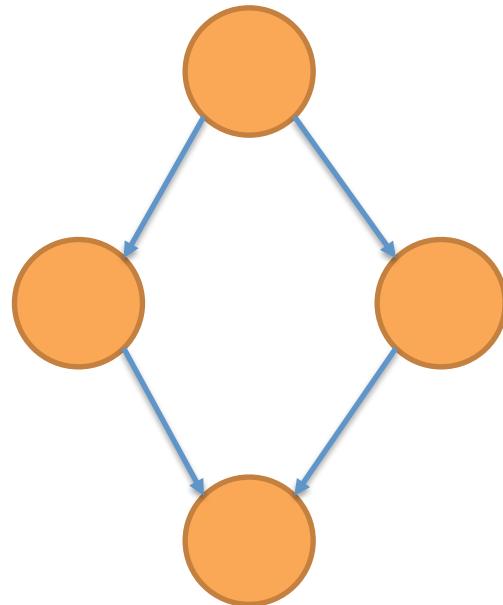
Windowing



- We may have to process data in time buckets, called windows
- Typically as part of some form of aggregation
- Windowing operations are available in the Kafka Stream DSL
- A window has a retention period to handle possible late arriving events

Parallel Stream Processing Pipelines

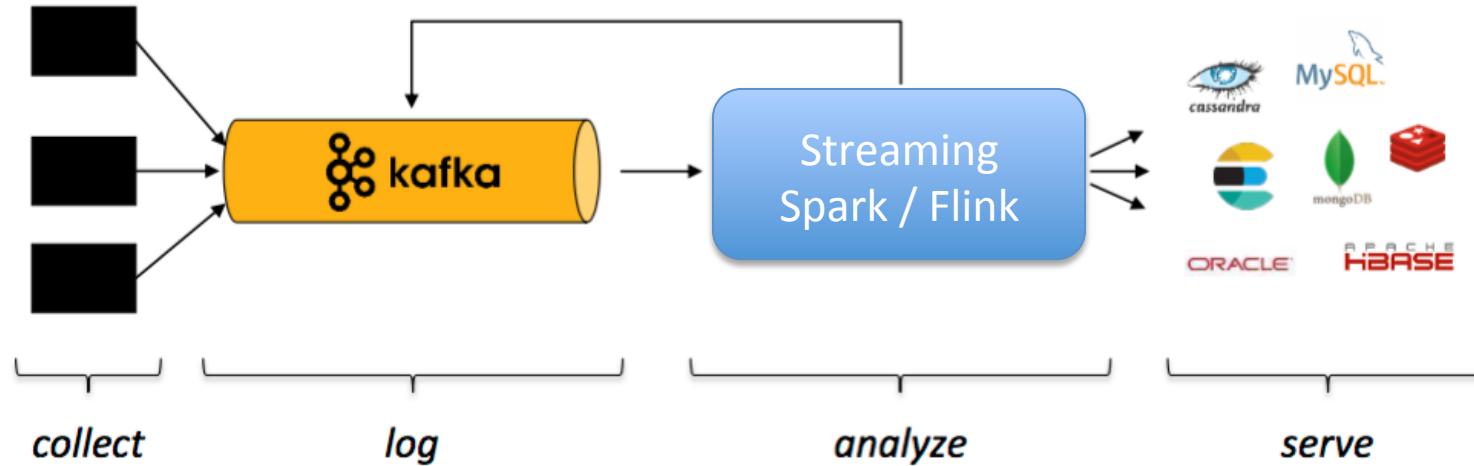
- The Kafka Stream DSL supports ways to parallelize and rendezvous processing steps
- Join
 - This operation merges two streams into a single stream
- Aggregation
 - Takes one input stream and yields a new stream by combining multiple input events into a single output event



Backpressure

- Kafka Streams don't need a backpressure mechanism
- Kafka uses a depth-first processing strategy
 - Each event consumed from Kafka goes through the complete processor topology before the next message is processed

Kafka with Spark or Flink



- Gather and backup streams
- Offer streams for consumption
- Provide stream recovery
- Analyze and correlate streams
- Create derived streams and state
- Provide these to downstream systems

Spark Streaming and Kafka

- We can set up Spark to use Kafka stream as the source of events
- Code structure:
 - A bit of configuration for Spark Streaming
 - A bit of configuration for Kafka parameters
 - Create stream from Kafka
- At this point, we have a DStream and we can do the normal Spark Streaming processing on it

Spark Streaming with Kafka – Spark Configuration

```
// Consume command line parameters
val Array(brokers, topics, interval) = args

// Create Spark configuration
val sparkConf = new SparkConf().setAppName("SparkKafka")

// Create streaming context, with batch duration in ms
val ssc = new StreamingContext(sparkConf, Duration(interval.toLong))
ssc.checkpoint("./output")
```

Spark Streaming with Kafka – Kafka Configuration

```
// Create a set of topics from a string
val topicsSet = topics.split(",").toSet

// Define Kafka parameters
val kafkaParams = Map[String, Object](
  "bootstrap.servers" -> brokers,
  "key.deserializer" -> classOf[StringDeserializer],
  "value.deserializer" -> classOf[StringDeserializer],
  "group.id" -> "use_a_separate_group_id_for_each_stream",
  "auto.offset.reset" -> "latest",
  "enable.auto.commit" -> (false: java.lang.Boolean))
```

Create a Kafka Stream in Spark and Use It

```
// Create a Kafka stream
val stream = KafkaUtils.createDirectStream[String, String](
  ssc, PreferConsistent, Subscribe[String, String](topicsSet,kafkaParams))
// Get messages - lines of text from Kafka
val lines = stream.map(consumerRecord => consumerRecord.value)
// Split lines into words
val words = lines.flatMap(_.split(" "))
// Map every word to a tuple
val wordMap = words.map(word => (word, 1))
// Count occurrences of each word
val wordCount = wordMap.reduceByKey(_ + _)
//Print the word count
wordCount.print()
```

Summary

- Kafka provides a stream processing library
- Two APIs
 - Low level API
 - Kafka Stream DSL
- The Kafka Stream DSL enables the definition of processing topologies
- Kafka streams support stateful stream processing
- Kafka has a reach support of functions and features
 - Windowing
 - Aggregation
 - Join
 - Functions (map, flatMap, etc.)
- Integration with other streaming systems: Spark, Flink, ...

Lab

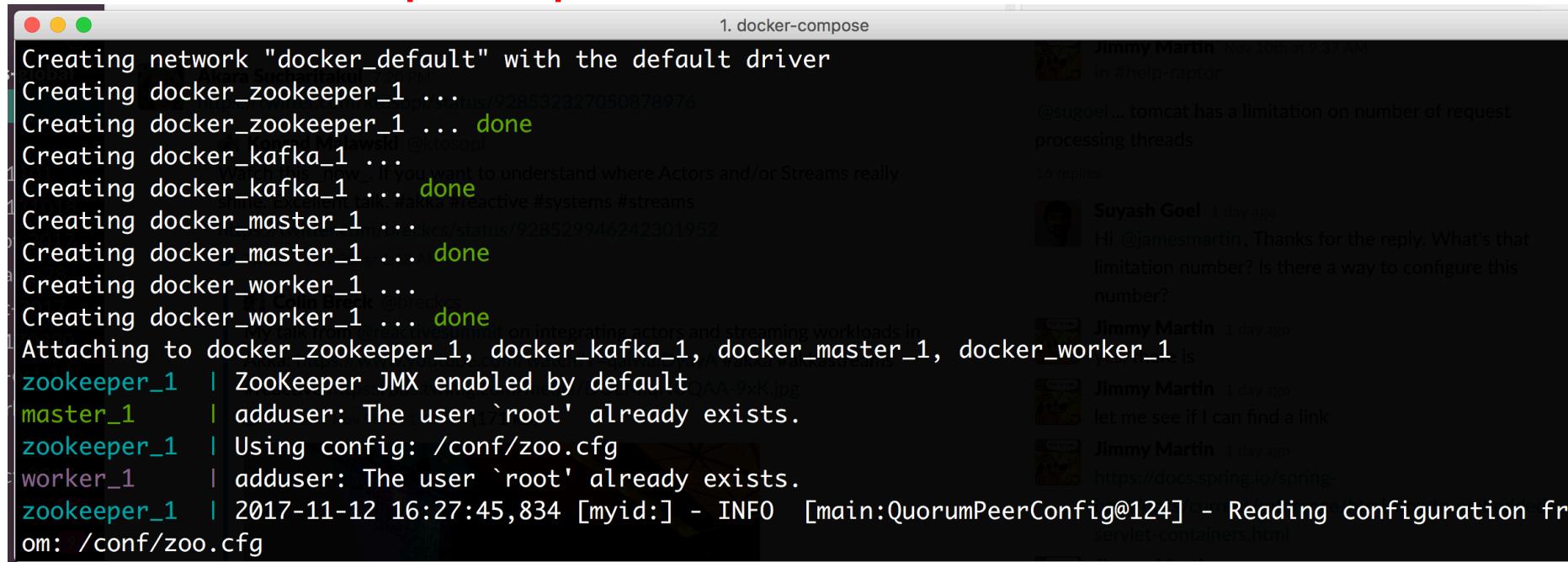
- We have 3 labs for streaming
- Lab 1: Word count using Kafka Streams
 - I will show this in slides
- Lab 2: Word count in Spark
 - I'll leave that for you to do after the class
- Lab 3: IoT example in Spark
 - I'll walk you through the code
- Example 4: IoT example implemented in Kafka Streams
 - I'll walk you through the code

Step 1: Setup Your docker.yml File

- We've done this before...
- The **docker-compose.yml** file is in the directory:
 - **kafka-lab/labs/06-Streaming/docker**
- Modify the **KAFKA_ADVERTISED_HOST_NAME** to refer to the IP of your machine
- Notice that in this docker-compose file we have more processes:
 - ZooKeeper
 - Kafka
 - Spark Master
 - Spark Worker

Step 2: Start Your Docker Processes

- Start your processes with:
docker-compose up



```
Creating network "docker_default" with the default driver
Creating docker_zookeeper_1 ...
Creating docker_zookeeper_1 ... done
Creating docker_kafka_1 ...
Creating docker_kafka_1 ... done
Creating docker_master_1 ...
Creating docker_master_1 ... done
Creating docker_worker_1 ...
Creating docker_worker_1 ... done
Attaching to docker_zookeeper_1, docker_kafka_1, docker_master_1, docker_worker_1
zookeeper_1  | ZooKeeper JMX enabled by default
master_1     | adduser: The user `root' already exists.
zookeeper_1  | Using config: /conf/zoo.cfg
worker_1     | adduser: The user `root' already exists.
zookeeper_1  | 2017-11-12 16:27:45,834 [myid:] - INFO  [main:QuorumPeerConfig@124] - Reading configuration from: /conf/zoo.cfg
```

Jimmy Martin Nov 10 at 9:57 AM
in #help-raptor
@sugoei... tomcat has a limitation on number of request processing threads
16 replies

Suyash Goel 1 day ago
Hi @jamesmartin , Thanks for the reply. What's that limitation number? Is there a way to configure this number?

Jimmy Martin 1 day ago
let me see if I can find a link
https://docs.spring.io/spring-servlet-containers.html

Step 3: Create Some Topics

- Start a new terminal (you may want to keep this terminal to eventually shutdown the docker containers)
- For the word-count we'll have two topics:
 - stream-input
 - stream-output
- Use the docker-compose exec to create the topics:

```
x sh
LM-SJN-21001415:docker pgraff$ docker-compose exec kafka /opt/kafka_2.11-0.10.1.1/bin/kafka-topics.sh --create --zookeeper zookeeper:2181 --replication-factor 1 --partitions 1 --topic stream-input
Created topic "stream-input".
LM-SJN-21001415:docker pgraff$ docker-compose exec kafka /opt/kafka_2.11-0.10.1.1/bin/kafka-topics.sh --create --zookeeper zookeeper:2181 --replication-factor 1 --partitions 1 --topic stream-output
Created topic "stream-output".
LM-SJN-21001415:docker pgraff$
```

Step 4: Create a Console Consumer

- Let's create a console consumer to listen to the end result
- In a separate terminal
 - Change to the docker directory
 - Run a console consumer listening to the stream-output:
**docker-compose exec kafka
/opt/kafka/bin/kafka-console-consumer.sh
--bootstrap-server localhost:9092
--topic stream-output
--from-beginning
--property print.key=true**

Prints the key as well as the value of each message

Where Are We?

The image shows a terminal window with the following text:

```
kafka_1    | [2017-11-12 16:38:57,810] INFO Created log for partition [__consumer_offsets,43] in /kafka/kafka-logs-0  
e43a4963505 with properties {compression.type -> producer, message.format.version -> 0.10.1-IV2, file.delete.delay.ms  
-> 60000, max.message.bytes -> 1000012, offset.commit.required.acks -> 1, timestamp.type -> CreateTime, min.insync.r  
eplicas -> 1, segment.jitter.ms -> 0, preallocate -> false, min.cleanable.dirty.ratio -> 0.5, index.interval.bytes ->  
4096, unclean.leader.election.enable -> true, retention.bytes -> -1, delete.retention.ms -> 86400000, cleanup.policy ->  
compact, flush.ms -> 2000, flush.offsets.interval.ms -> 1000000, flush.size.bytes -> 10485760, flush.size.requests -> 10000,  
message.timestamp.difference.max.ms -> 9223372036854775807, segment.index.bytes -> 1048576, flush.messages -> 9  
223372036854775807}. (kafka.log.LogManager)
```

Below the terminal, a LinkedIn post by Jimmy Martin is visible:

docker containers (kafka, zookeeper, master, worker)

Excellent talk #akka #reactive #systems #streams

LM-SJN-21001415:docker pgraff\$ docker-compose exec kafka /opt/kafka_2.11-0.10.1.1/bin/kafka-console-consumer.sh --boot
strap-server localhost:9092 --topic stream-output --from-beginning --property print.key=true

My talk from @reactivesummit on Integrating actors and streaming workloads in
Akka: <https://www.youtube.com/watch?v=qIwaiDyayA> #akka #akkastreams

Jimmy Martin 1 day ago
yes, there is

console-consumer listening to stream-output



Step 5: Create a Console Producer

- We also need to be able to produce the input data
- Open a terminal and change directory to the docker directory
- Run the console-producer:

```
docker-compose exec kafka  
/opt/kafka/bin/kafka-console-producer.sh  
--broker-list kafka:9092  
--topic stream-input
```

Where Are We?

The image displays three terminal windows side-by-side, each showing logs from a Kafka Docker container. Overlaid on these logs are large, semi-transparent red text labels identifying the components and their functions.

- Top Terminal:** Shows logs from the Kafka log manager. The text is mostly obscured by a large red watermark containing the text "docker containers (kafka, zookeeper, master, worker)".
- Middle Terminal:** Shows logs from a Kafka broker. It includes the following text:

```
kafka_1 | [2017-11-12 16:38:57,810] INFO Created log for partition [<__consumer_offsets,43] in /kafka/kafka-logs-0  
e43a4963505 with properties {compression.type -> producer, message.format.version -> 0.10.1-IV2, file.delete.delay.ms  
-> 60000, max.message.bytes -> 1000012, min.compaction.lag.ms -> 0, message.timestamp.type -> CreateTime, min.insync.r  
eplicas -> 1, segment.jitter.ms -> 0, perPartition.unclean.leader.election.enable -> false, segment.ratio -> 0.5, index.interval.bytes ->  
4096, unclean.leader.election.enable -> true, retention.bytes -> -1, delete.retention.ms -> 86400000, cleanup.policy ->  
> compact, flush.ms -> 9223372036854775807, segment.ms -> 604800000, segment.bytes -> 104857600, retention.ms -> 60480  
0000, message.timestamp.type -> CreateTime, max.message.bytes -> 1000012, unclean.leader.election.enable -> true, flush.ms -> 9  
223372036854775807}. (kafka.log.LogManager)
```
- Bottom Terminal:** Shows logs from a Kafka producer and consumer. The producer log includes:

```
LM-SJN-21001415:06-Streaming pgraff$ cd docker/  
LM-SJN-21001415:06-Streaming pgraff$ docker exec kafka /  
opt/kafka_2.11-0.10.1.1/bin/kafka-console-producer.sh --br  
oker-list kafka:9092 --topic stream-input
```

And the consumer log includes:

```
LM-SJN-21001415:docker pgraff$ docker-compose exec kafka /  
opt/kafka_2.11-0.10.1.1/bin/kafka-console-consumer.sh --bo  
otstrap-server kafka:9092 --topic stream-output --from  
-beginning --property print.key=true
```

Step 6: Build and Run the Stream Processor

- Open a terminal and change directory to the **wordcount-kafka-solution**
- Run **mvn package** and after that **target/wordcounter**

The screenshot shows a terminal window with the following output:

```
[INFO] BUILD SUCCESS
[INFO] -----LM-SJN-21001415:wordcount-kafka-solution pgraff$ target/wordcounter
[INFO] Total time: 3.121 s
[INFO] Finished at: 2017-11-12T10:55:39-06:00
[INFO] Final Memory: 44M/365M
[INFO] -----
LM-SJN-21001415:wordcount-kafka-solution pgraff$ target/wordcounter
log4j:WARN No appenders could be found for logger (org.apache.kafka.streams.StreamsConfig).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
Press enter to quit the stream processor
```

The terminal window has a title bar "2.java". The background of the terminal shows other command-line history related to Docker and Kafka.

Where Are We?

```
for partition [__consumer_offsets,18] in /kafka/kafka-logs  
-0e43a4963505 with properties {compression.type -> produce  
r, message.format.version -> 0.10.1-IV2, file.delete.delay  
.ms -> 60000, max.message.bytes -> 1000012, min.compaction  
.lag.ms -> 0, min.insync.replicas -> 1, segment.jitter.ms -> 0, preallocate ->  
false, min.cleanable.dirty.ratio -> 0.5, index.interval.b  
ytes -> 4096, unclean.leader.election.enable -> true, rete  
ntion.bytes -> -1, delete.retention.ms -> 86400000, cleanu  
p.policy -> compact, flush.ms -> 9223372036854775807, segm
```

docker containers

```
LM-SJN-21001415:06-Streaming pgraff$ cd docker/  
LM-SJN-21001415:docker pgraff$ docker-compose exec kafka /  
opt/kafka_2.11-0.10.1.1/bin/kafka-console-producer.sh --br  
oker-list kafka:9092 --topic stream-input
```

console-producer
producing to
stream-input

1.java

x java

```
-----  
LM-SJN-21001415:wordcount-kafka-solution pgraff$ target/wor  
dcounter  
log4j:WARN No appenders could be found for logger (org.apac  
he.kafka.streams.processor.internals)  
log4j:WARN Please initialize the log4j system properly.  
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html  
#noconfig for more info.  
Press enter to quit the stream processor
```

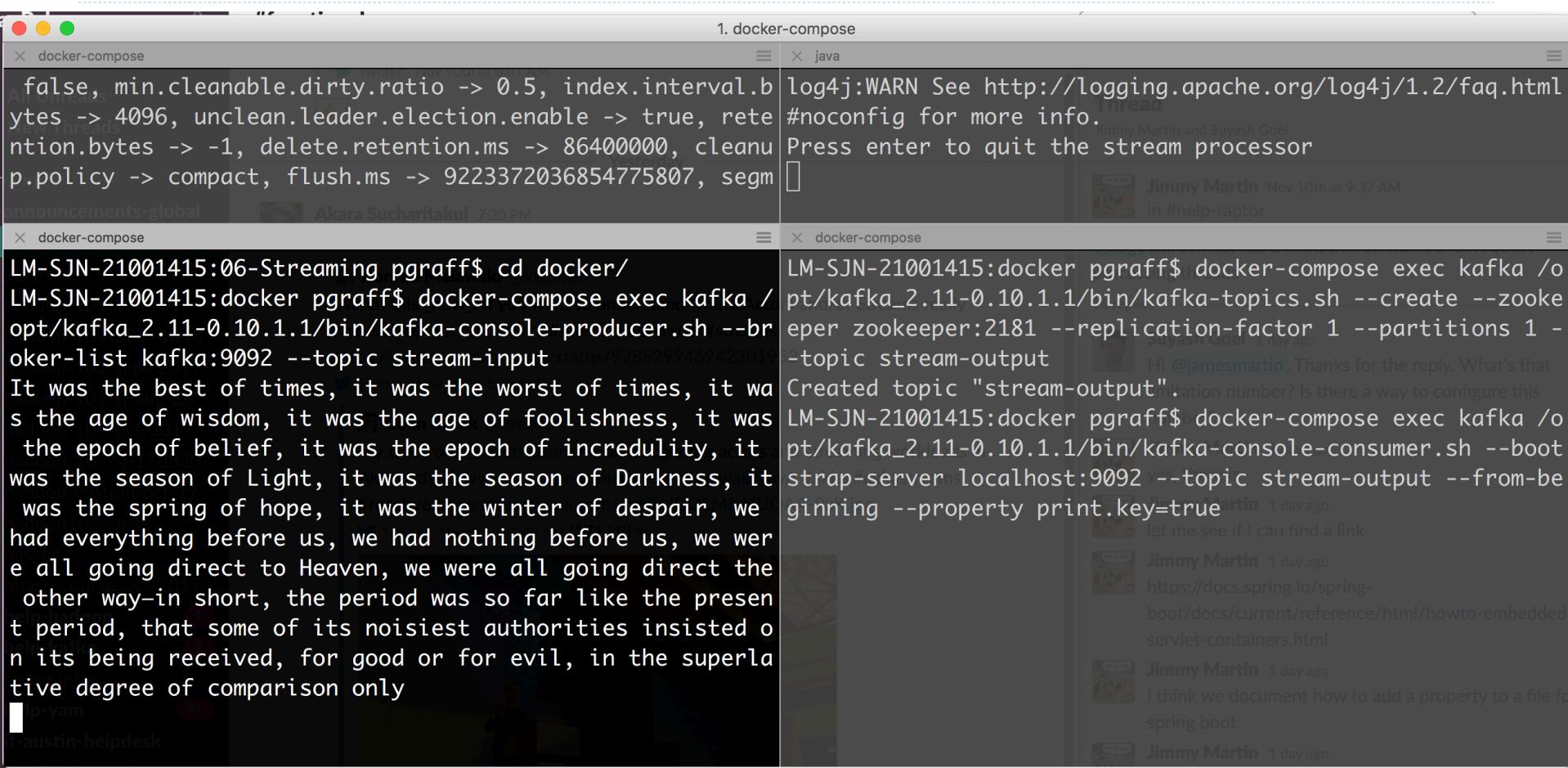
The event processor

x docker-compose

```
LM-SJN-21001415:docker pgraff$ docker-compose exec kafka /  
opt/kafka_2.11-0.10.1.1/bin/kafka-console-consumer.sh --boot  
strap-server localhost:9092 --topic stream-output --from-be  
ginning --property print.key=true
```

console-consumer
listening to
stream-output

Step 7: Enter Some Text in the Producer



The screenshot shows a terminal window with two tabs open. The left tab, titled 'docker-compose', contains the command:

```
false, min.cleanable.dirty.ratio -> 0.5, index.interval.bytes -> 4096, unclean.leader.election.enable -> true, retention.bytes -> -1, delete.retention.ms -> 86400000, cleanup.policy -> compact, flush.ms -> 9223372036854775807, segm
```

The right tab, titled 'java', shows the log output of a Java application using Log4j:

```
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html  
#noconfig for more info.  
Press enter to quit the stream processor
```

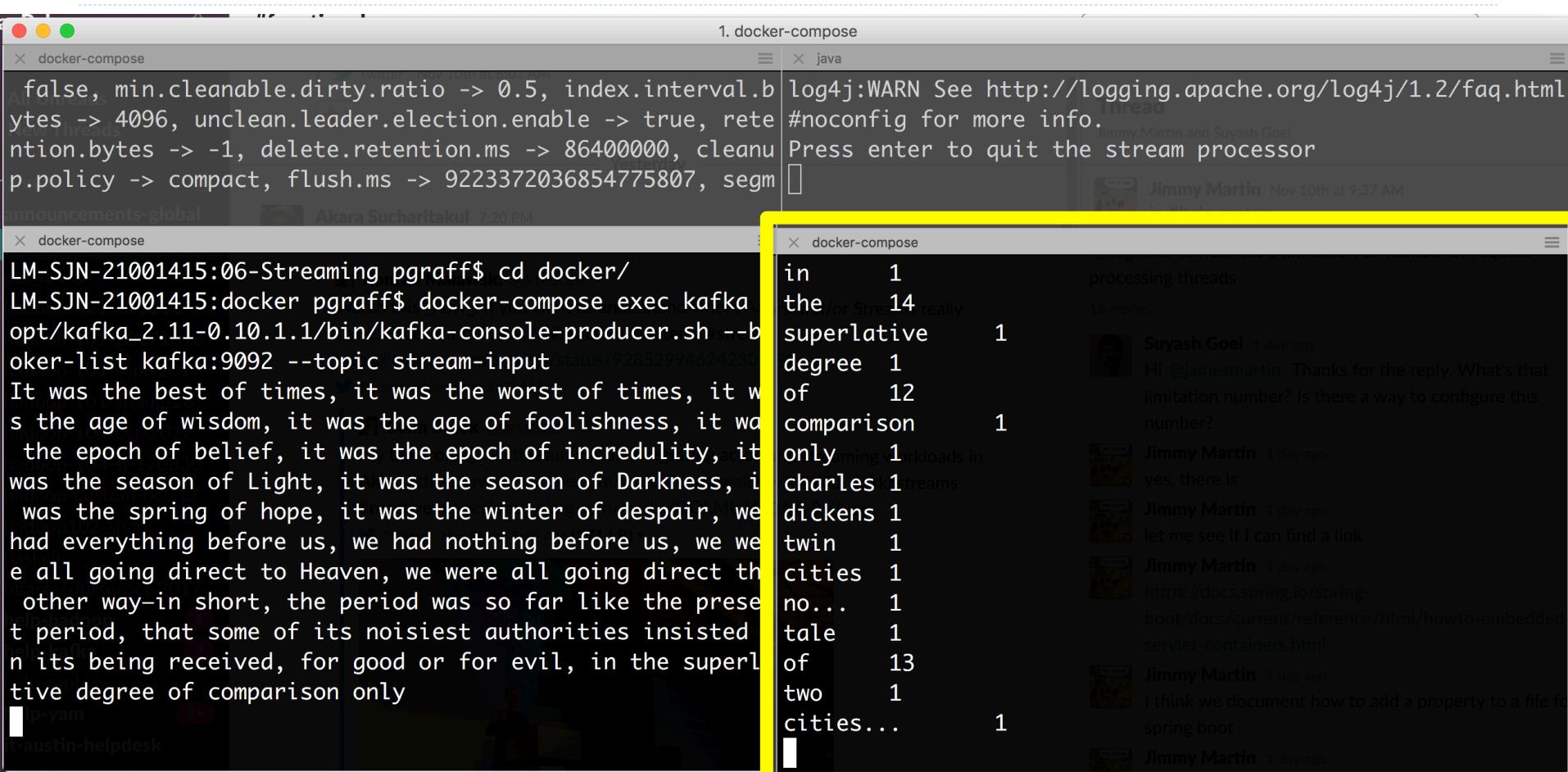
Below the terminal window, there is a message from Jimmy Martin in a Slack channel:

Nov 10th at 9:37 AM
in #help-raptor

LM-SJN-21001415:06-Streaming pgraff\$ cd docker/
LM-SJN-21001415:docker pgraff\$ docker-compose exec kafka /opt/kafka_2.11-0.10.1.1/bin/kafka-console-producer.sh --broker-list kafka:9092 --topic stream-input
It was the best of times, it was the worst of times, it was the age of wisdom, it was the age of foolishness, it was the epoch of belief, it was the epoch of incredulity, it was the season of Light, it was the season of Darkness, it was the spring of hope, it was the winter of despair, we had everything before us, we had nothing before us, we were all going direct to Heaven, we were all going direct the other way—in short, the period was so far like the present period, that some of its noisiest authorities insisted on its being received, for good or for evil, in the superlative degree of comparison only

LM-SJN-21001415:docker pgraff\$ docker-compose exec kafka /opt/kafka_2.11-0.10.1.1/bin/kafka-topics.sh --create --zookeeper zookeeper:2181 --replication-factor 1 --partitions 1 --topic stream-output
Created topic "stream-output".
LM-SJN-21001415:docker pgraff\$ docker-compose exec kafka /opt/kafka_2.11-0.10.1.1/bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic stream-output --from-beginning --property print.key=true
Martin 1 day ago
let me see if I can find a link
Jimmy Martin 1 day ago
<https://docs.spring.io/spring-boot/docs/current/reference/html/howto-embedded-servlet-containers.html>
Jimmy Martin 1 day ago
I think we document how to add a property to a file for spring boot
Jimmy Martin 1 day ago

After some time...



```
false, min.cleanable.dirty.ratio -> 0.5, index.interval.bytes -> 4096, unclean.leader.election.enable -> true, retention.bytes -> -1, delete.retention.ms -> 86400000, cleanup.policy -> compact, flush.ms -> 9223372036854775807, segm  
announcements-global  
Akara Sucharitakul 7:20 PM  
LM-SJN-21001415:06-Streaming pgraff$ cd docker/  
LM-SJN-21001415:docker pgraff$ docker-compose exec kafka  
opt/kafka_2.11-0.10.1.1/bin/kafka-console-producer.sh --b  
oker-list kafka:9092 --topic stream-input status/92852994624230  
It was the best of times, it was the worst of times, it w  
s the age of wisdom, it was the age of foolishness, it wa  
the epoch of belief, it was the epoch of incredulity, it  
was the season of Light, it was the season of Darkness, i  
was the spring of hope, it was the winter of despair, we  
had everything before us, we had nothing before us, we we  
e all going direct to Heaven, we were all going direct th  
other way—in short, the period was so far like the prese  
nt period, that some of its noisiest authorities insisted  
on its being received, for good or for evil, in the superl  
ative degree of comparison only
```

Word	Count
in	1
the	14
superlative	1
degree	1
of	12
comparison	1
only	1
charles	1
dickens	1
twin	1
cities	1
no...	1
tale	1
of	13
two	1
cities...	1

log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html
#noconfig for more info.
Press enter to quit the stream processor

Jimmy Martin and Suyash Goel
Jimmy Martin Nov 10th at 9:37 AM

processing threads
16 replies

Suyash Goel 1 day ago
Hi @jamesmartin, Thanks for the reply. What's that
limitation number? Is there a way to configure this
number?

Jimmy Martin 1 day ago
yes, there is

Jimmy Martin 1 day ago
let me see if I can find a link

Jimmy Martin 1 day ago
<https://docs.spring.io/spring-boot/docs/current/reference/html/howto-embedded-servlet-containers.html>

Jimmy Martin 1 day ago
I think we document how to add a property to a file in
spring boot

Jimmy Martin 1 day ago

Lab/Example: Another Problem

Devices



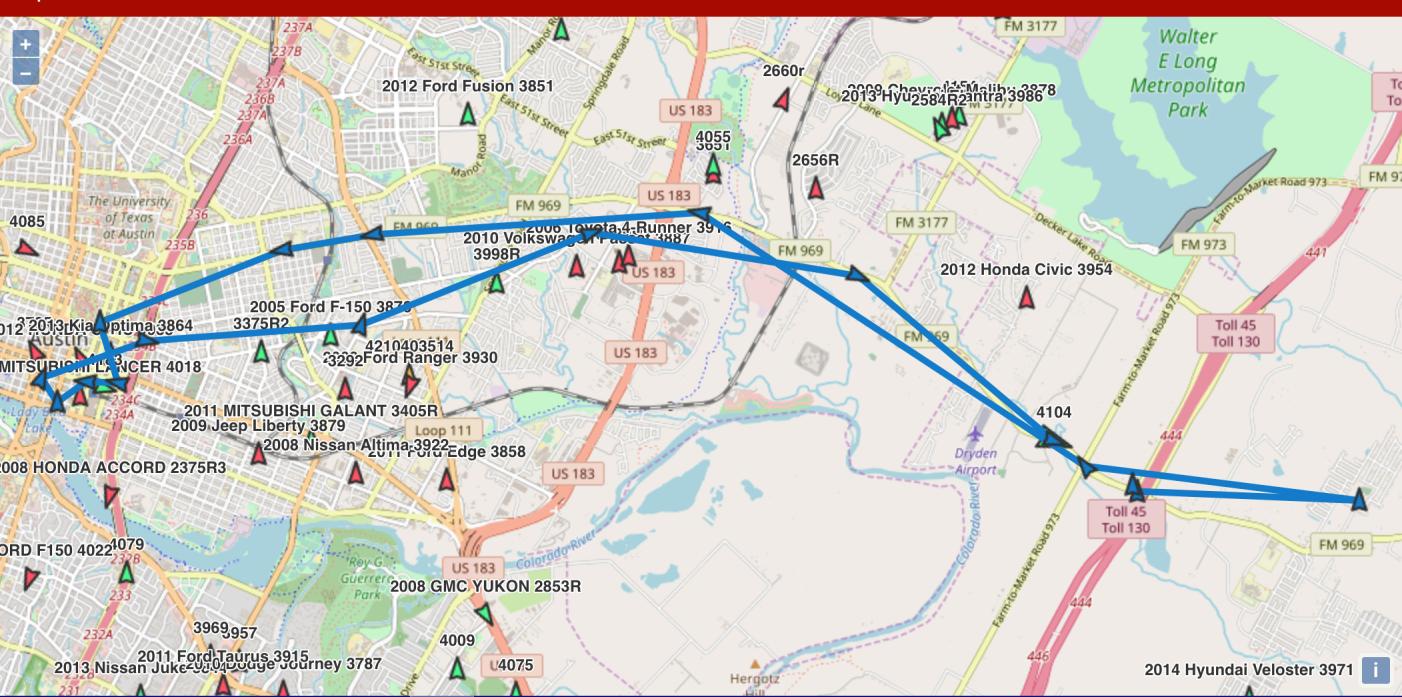
Name ↑	Last Update
2005 CHEVY AVALANC...	2017-11-12 07:52:57
2005 Volkswagen Golf 3918	2017-11-12 09:28:18
2007 BUICK LACROSS...	2017-11-12 10:20:46
2007 CHEVY SUBURBA...	2017-11-12 10:19:07
2007 FORD EXPLORER...	2017-11-12 07:21:23

Group Search

State

Attribute ↑	Value
Address	display
Battery	No
Color	BLACK
Course	E
Ignition	Yes
Ip	80.65.250.116
Latitude	30.259717
Longitude	-97.615567

Map



Device | From To | Show Clear

Event	Time	Latitude	Longitude	Speed	Address
.. moving	2017-11-12 01:22:00	30.259717	-97.615567	50.6 mph	display
.. moving	2017-11-12 01:17:00	30.278067	-97.640783	51.8 mph	display
.. moving	2017-11-12 01:12:00	30.282883	-97.675117	31.1 mph	display
.. moving	2017-11-12 01:07:00	30.272633	-97.704783	12.7 mph	display
.. moving	2017-11-12 01:02:00	30.270833	-97.732417	16.1 mph	display
.. moving	2017-11-12 00:57:00	30.266733	-97.745950	20.7 mph	display
.. moving	2017-11-12 00:52:00	30.264017	-97.743933	2.3 mph	display

Example Description

- We get a stream of vehicle locations
- Each location contains:
 - Vehicle ID
 - Data of observation
 - Speed
 - Direction
 - Latitude
 - Longitude

Problem and Plan

- We want to know where the vehicle is most likely to be parked
- Our plan
 - 1. Parse the vehicle data from an input stream
 - 2. Filter out the only those locates where the vehicle was at rest
 - 3. Convert the vehicle locate into keys that we can accumulate. The key would be based on
 - Vehicle ID
 - Geo location
 - 4. Group the locates together by key and count them
 - 5. Stream the frequent parking count to another stream

Input File: vehicle_1_to_1000.tsv

678 120 2016-08-16 16:31:21 30 162 -97.796833333334 30.46651666666666/
679 120 2016-08-16 16:31:31 5 159 -97.79646666666666 30.465716666666665
680 113 2016-08-16 16:31:37 5 159 -97.79645 30.465733333333333
681 120 2016-08-16 16:31:41 5 159 -97.79646666666666 30.465716666666665
682 120 2016-08-16 16:31:51 5 159 -97.79646666666666 30.465716666666665
683 120 2016-08-16 16:32:01 14 154 -97.7963833333334 30.465566666666668
684 120 2016-08-16 16:32:11 22 163 -97.79596666666667 30.46458333333334
685 120 2016-08-16 16:32:17 28.4 159 -97.795695 30.463865
686 120 2016-08-16 16:32:21 26 159 -97.7954833333334 30.463366666666666
687 107 2016-08-16 16:32:29 25 159 -97.7950833333334 30.4624
688 120 2016-08-16 16:32:31 24 157 -97.795 30.462216666666666
689 120 2016-08-16 16:32:41 7 160 -97.79466666666667 30.4615
690 120 2016-08-16 16:32:51 2 158 -97.7946333333334 30.46143333333332
691 113 2016-08-16 16:32:52 2.1 160 -97.79464166666666 30.46143333333332
692 120 2016-08-16 16:33:01 2 158 -97.7946333333334 30.46143333333332
693 111 2016-08-16 16:33:08 2 161 -97.7946833333334 30.46145
694 120 2016-08-16 16:33:11 2 159 07 70463222222224 30.46143222222222

Message Pump: Simulator

- We've created a message pump application to simulate the vehicle input
- You can find this implementation in the directory:
kafka-lab/labs/06-Streaming/iot-kafka-solution/gps-pump
- The implementation reads the TSV file and produces messages into a topic called:
gps-locations
- In the real world, we would have to write some edge servers that receives the data from the vehicles and publish the topic into the topic

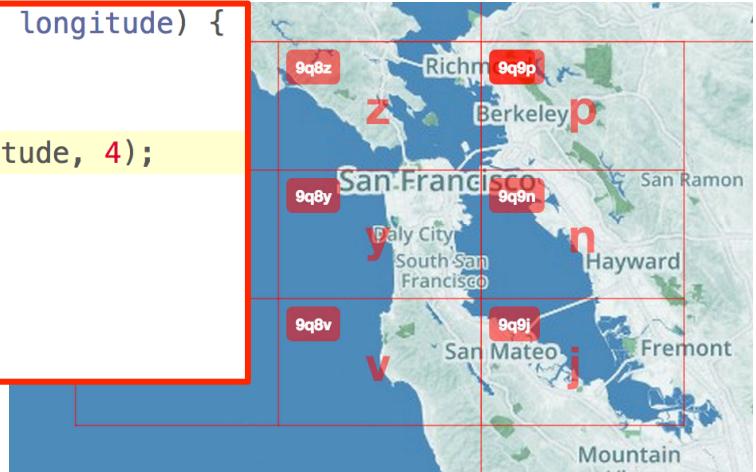
Event Processor

- We've created an event processor based on kafka-streaming (and another on Spark that we'll look at later)
- You can find this implementation in the directory:
kafka-lab/labs/06-Streaming/iot-kafka-solution/processor
- This processor uses kafka-streams to read the events from the topic **gps-locations** and produces messages on the topic **frequent-parking**
 - The messages produced:
 - Key: Vehicle and location key based on GeoHash
 - Value: parking count

GeoHash

- We build a key based on the vehicle id and a GeoHash
- GeoHash allows us to focus on an area instead of absolute locates

```
9  public LocationKey(String id, double latitude, double longitude) {  
10     this.objectId = id;  
11     try {  
12         this.geoHash = GeoHash.encodeHash(longitude, latitude, 4);  
13     }  
14     catch (Exception e) {  
15         System.out.println(e.getMessage());  
16     }  
17 }
```



Kafka Stream Solution

```
1 source
2   .map( (key,value) -> new KeyValue<>(key, value.split("\t")))
3   .filter((key,value)--> value.length > 5 && Double.parseDouble(value[2]) == 0)
4   .map((key,value) ->
5     new KeyValue<>(
6       new LocationKey(value[0],
7         Double.parseDouble(value[4]),
8         Double.parseDouble(value[5])).toString(),
9         value[0]))
10  .groupByKey()
11  .count("parking")
12  .mapValues((value) -> Long.toString(value))
13  .to("frequent-parking");
```

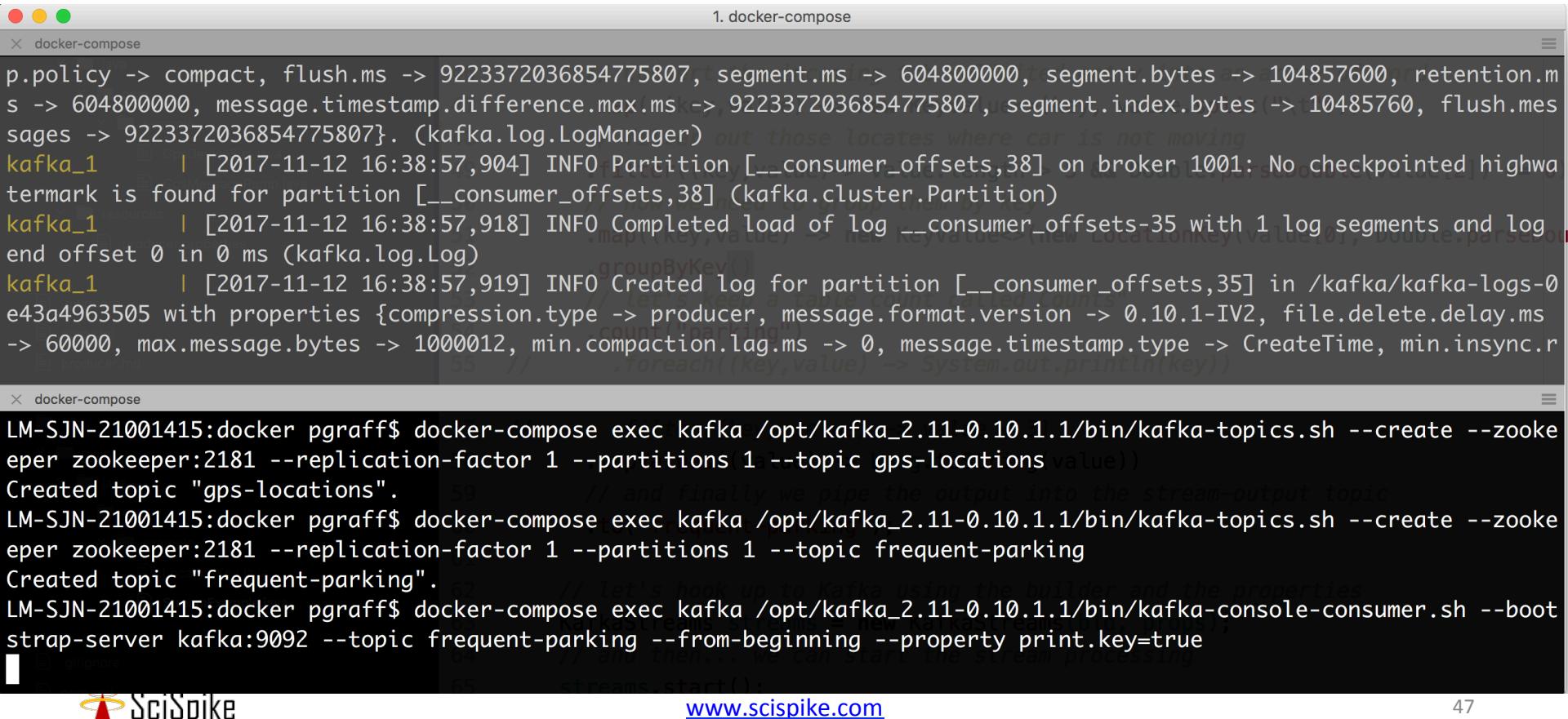
Run Solution in Kafka

1. Modify the docker-compose file to use your (which you may already have done in the word-count exercise)
2. Create the topics:
 - gps-locations
 - frequent-parking
3. Run a kafka-consumer to listen to the frequent-parking topic (with the show key property)
4. Build and run the message pump
5. Build and run the processor
6. Wait to see the parking counts...

Creating the Topics and Start the Console Consumer

```
1. docker-compose
policy -> compact, flush.ms -> 9223372036854775807, segment.ms -> 604800000, segment.bytes -> 104857600, retention.ms -> 604800000, message.timestamp.difference.max.ms -> 9223372036854775807, segment.index.bytes (-> 10485760, flush.messages -> 9223372036854775807}. (kafka.log.LogManager)
kafka_1    | [2017-11-12 16:38:57,904] INFO Partition [<__consumer_offsets,38] on broker 1001: No checkpointed highwatermark is found for partition [<__consumer_offsets,38] (kafka.cluster.Partition)
kafka_1    | [2017-11-12 16:38:57,918] INFO Completed load of log __consumer_offsets-35 with 1 log segments and log end offset 0 in 0 ms (kafka.log.Log)
kafka_1    | [2017-11-12 16:38:57,919] INFO Created log for partition [<__consumer_offsets,35] in /kafka/kafka-logs-0e43a4963505 with properties {compression.type -> producer, message.format.version -> 0.10.1-IV2, file.delete.delay.ms -> 60000, max.message.bytes -> 1000012, min.compaction.lag.ms -> 0, message.timestamp.type -> CreateTime, min.insync.r

LM-SJN-21001415:docker pgraff$ docker-compose exec kafka /opt/kafka_2.11-0.10.1.1/bin/kafka-topics.sh --create --zookeeper zookeeper:2181 --replication-factor 1 --partitions 1 --topic gps-locations
Created topic "gps-locations".
LM-SJN-21001415:docker pgraff$ docker-compose exec kafka /opt/kafka_2.11-0.10.1.1/bin/kafka-topics.sh --create --zookeeper zookeeper:2181 --replication-factor 1 --partitions 1 --topic frequent-parking
Created topic "frequent-parking".
LM-SJN-21001415:docker pgraff$ docker-compose exec kafka /opt/kafka_2.11-0.10.1.1/bin/kafka-console-consumer.sh --bootstrap-server kafka:9092 --topic frequent-parking --from-beginning --property print.key=true



The image shows a terminal window with two tabs: 'docker-compose' and '1. docker-compose'. The '1. docker-compose' tab displays Kafka logs from brokers kafka_1 and kafka_2. The logs show the initial state of the __consumer_offsets topic, including the loading of log segments and the creation of a new log for partition 35. The 'docker-compose' tab shows the execution of three commands to create topics 'gps-locations' and 'frequent-parking', and then start a console consumer for the 'frequent-parking' topic.


```

Build and Start the Message Pump

```
sent: 22      2016-08-19 12:11:38    11      13      31      33      35      37      39      41      43      45      47      49      51      53      55      57      59      61      63      65      67      69      71      73      75      77      79      81      83      85      87      89      91      93      95      97      99      101      103      105      107      109      111      113      115      117      119      121      123      125      127      129      131      133      135      137      139      141      143      145      147      149      151      153      155      157      159      161      163      165      167      169      171      173      175      177      179      181      183      185      187      189      191      193      195      197      199      201      203      205      207      209      211      213      215      217      219      221      223      225      227      229      231      233      235      237      239      241      243      245      247      249      251      253      255      257      259      261      263      265      267      269      271      273      275      277      279      281      283      285      287      289      291      293      295      297      299      301      303      305      307      309      311      313      315      317      319      321      323      325      327      329      331      333      335      337      339      341      343      345      347      349      351      353      355      357      359      361      363      365      367      369      371      373      375      377      379      381      383      385      387      389      391      393      395      397      399      401      403      405      407      409      411      413      415      417      419      421      423      425      427      429      431      433      435      437      439      441      443      445      447      449      451      453      455      457      459      461      463      465      467      469      471      473      475      477      479      481      483      485      487      489      491      493      495      497      499      501      503      505      507      509      511      513      515      517      519      521      523      525      527      529      531      533      535      537      539      541      543      545      547      549      551      553      555      557      559      561      563      565      567      569      571      573      575      577      579      581      583      585      587      589      591      593      595      597      599      601      603      605      607      609      611      613      615      617      619      621      623      625      627      629      631      633      635      637      639      641      643      645      647      649      651      653      655      657      659      661      663      665      667      669      671      673      675      677      679      681      683      685      687      689      691      693      695      697      699      701      703      705      707      709      711      713      715      717      719      721      723      725      727      729      731      733      735      737      739      741      743      745      747      749      751      753      755      757      759      761      763      765      767      769      771      773      775      777      779      781      783      785      787      789      791      793      795      797      799      801      803      805      807      809      811      813      815      817      819      821      823      825      827      829      831      833      835      837      839      841      843      845      847      849      851      853      855      857      859      861      863      865      867      869      871      873      875      877      879      881      883      885      887      889      891      893      895      897      899      901      903      905      907      909      911      913      915      917      919      921      923      925      927      929      931      933      935      937      939      941      943      945      947      949      951      953      955      957      959      961      963      965      967      969      971      973      975      977      979      981      983      985      987      989      991      993      995      997      999      1001      1003      1005      1007      1009      1011      1013      1015      1017      1019      1021      1023      1025      1027      1029      1031      1033      1035      1037      1039      1041      1043      1045      1047      1049      1051      1053      1055      1057      1059      1061      1063      1065      1067      1069      1071      1073      1075      1077      1079      1081      1083      1085      1087      1089      1091      1093      1095      1097      1099      1101      1103      1105      1107      1109      1111      1113      1115      1117      1119      1121      1123      1125      1127      1129      1131      1133      1135      1137      1139      1141      1143      1145      1147      1149      1151      1153      1155      1157      1159      1161      1163      1165      1167      1169      1171      1173      1175      1177      1179      1181      1183      1185      1187      1189      1191      1193      1195      1197      1199      1201      1203      1205      1207      1209      1211      1213      1215      1217      1219      1221      1223      1225      1227      1229      1231      1233      1235      1237      1239      1241      1243      1245      1247      1249      1251      1253      1255      1257      1259      1261      1263      1265      1267      1269      1271      1273      1275      1277      1279      1281      1283      1285      1287      1289      1291      1293      1295      1297      1299      1301      1303      1305      1307      1309      1311      1313      1315      1317      1319      1321      1323      1325      1327      1329      1331      1333      1335      1337      1339      1341      1343      1345      1347      1349      1351      1353      1355      1357      1359      1361      1363      1365      1367      1369      1371      1373      1375      1377      1379      1381      1383      1385      1387      1389      1391      1393      1395      1397      1399      1401      1403      1405      1407      1409      1411      1413      1415      1417      1419      1421      1423      1425      1427      1429      1431      1433      1435      1437      1439      1441      1443      1445      1447      1449      1451      1453      1455      1457      1459      1461      1463      1465      1467      1469      1471      1473      1475      1477      1479      1481      1483      1485      1487      1489      1491      1493      1495      1497      1499      1501      1503      1505      1507      1509      1511      1513      1515      1517      1519      1521      1523      1525      1527      1529      1531      1533      1535      1537      1539      1541      1543      1545      1547      1549      1551      1553      1555      1557      1559      1561      1563      1565      1567      1569      1571      1573      1575      1577      1579      1581      1583      1585      1587      1589      1591      1593      1595      1597      1599      1601      1603      1605      1607      1609      1611      1613      1615      1617      1619      1621      1623      1625      1627      1629      1631      1633      1635      1637      1639      1641      1643      1645      1647      1649      1651      1653      1655      1657      1659      1661      1663      1665      1667      1669      1671      1673      1675      1677      1679      1681      1683      1685      1687      1689      1691      1693      1695      1697      1699      1701      1703      1705      1707      1709      1711      1713      1715      1717      1719      1721      1723      1725      1727      1729      1731      1733      1735      1737      1739      1741      1743      1745      1747      1749      1751      1753      1755      1757      1759      1761      1763      1765      1767      1769      1771      1773      1775      1777      1779      1781      1783      1785      1787      1789      1791      1793      1795      1797      1799      1801      1803      1805      1807      1809      1811      1813      1815      1817      1819      1821      1823      1825      1827      1829      1831      1833      1835      1837      1839      1841      1843      1845      1847      1849      1851      1853      1855      1857      1859      1861      1863      1865      1867      1869      1871      1873      1875      1877      1879      1881      1883      1885      1887      1889      1891      1893      1895      1897      1899      1901      1903      1905      1907      1909      1911      1913      1915      1917      1919      1921      1923      1925      1927      1929      1931      1933      1935      1937      1939      1941      1943      1945      1947      1949      1951      1953      1955      1957      1959      1961      1963      1965      1967      1969      1971      1973      1975      1977      1979      1981      1983      1985      1987      1989      1991      1993      1995      1997      1999      2001      2003      2005      2007      2009      2011      2013      2015      2017      2019      2021      2023      2025      2027      2029      2031      2033      2035      2037      2039      2041      2043      2045      2047      2049      2051      2053      2055      2057      2059      2061      2063      2065      2067      2069      2071      2073      2075      2077      2079      2081      2083      2085      2087      2089      2091      2093      2095      2097      2099      2101      2103      2105      2107      2109      2111      2113      2115      2117      2119      2121      2123      2125      2127      2129      2131      2133      2135      2137      2139      2141      2143      2145      2147      2149      2151      2153      2155      2157      2159      2161      2163      2165      2167      2169      2171      2173      2175      2177      2179      2181      2183      2185      2187      2189      2191      2193      2195      2197      2199      2201      2203      2205      2207      2209      2211      2213      2215      2217      2219      2221      2223      2225      2227      2229      2231      2233      2235      2237      2239      2241      2243      2245      2247      2249      2251      2253      2255      2257      2259      2261      2263      2265      2267      2269      2271      2273      2275      2277      2279      2281      2283      2285      2287      2289      2291      2293      2295      2297      2299      2301      2303      2305      2307      2309      2311      2313      2315      2317      2319      2321      2323      2325      2327      2329      2331      2333      2335      2337      2339      2341      2343      2345      2347      2349      2351      2353      2355      2357      2359      2361      2363      2365      2367      2369      2371      2373      2375      2377      2379      2381      2383      2385      2387      2389      2391      2393      2395      2397      2399      2401      2403      2405      2407      2409      2411      2413      2415      2417      2419      2421      2423      2425      2427      2429      2431      2433      2435      2437      2439      2441      2443      2445      2447      2449      2451      2453      2455      2457      2459      2461      2463      2465      2467      2469      2471      2473      2475      2477      2479      2481      2483      2485      2487      2489      2491      2493      2495      2497      2499      2501      2503      2505      2507      2509      2511      2513      2515      2517      2519      2521      2523      2525      2527      2529      2531      2533      2535      2537      2539      2541      2543      2545      2547      2549      2551      2553      2555      2557      2559      2561      2563      2565      2567      2569      2571      2573      2575      2577      2579      2581      2583      2585      2587      2589      2591      2593      2595      2597      2599      2601      2603      2605      2607      2609      2611      2613      2615      2617      2619      2621      2623      2625      2627      2629      2631      2633      2635      2637      2639      2641      2643      2645      2647      2649      2651      2653      2655      2657      2659      2661      2663      2665      2667      2669      2671      2673      2675      2677      2679      2681      2683      2685      2687      2689      2691      2693      2695      2697      2699      2701      2703      2705      2707      2709      2711      2713      2715      2717      2719      2721      2723      2725      2727      2729      2731      2733      2735      2737      2739      2741      2743      2745      2747      2749      2751      2753      2755      2757      2759      2761      2763      2765      2767      2769      2771      2773      2775      2777      2779      2781      2783      2785      2787      2789      2791      2793      2795      2797      2799      2801      2803      2805      2807      2809      2811      2813      2815      2817      2819      2821      2823      2825      2827      2829      2831      2833      2835      2837      2839      2841      2843      2845      2847      2849      2851      2853      2855      2857      2859      2861      2863      2865      2867      2869      2871      2873      2875      2877      2879      2881      2883      2885      2887      2889      2891      2893      2895      2897      2899      2901      2903      2905      2907      2909      2911      2913      2915      2917      2919      2921      2923      2925      2927      2929      2931      2933      2935      2937      2939      2941      2943      2945      2947      2949      2951      2953      2955      2957      2959      2961      2963      2965      2967      2969      2971      2973      2975      2977      2979      2981      2983      2985      2987      2989      2991      2993      2995      2997      2999      3001      3003      3005      3007      3009      3011      3013      3015      3017      3019      3021      3023      3025      3027      3029      3031      3033      3035      3037      3039      3041      3043      3045      3047      3049      3051      3053      3055      3057      3059      3061      3063      3065      3067      3069      3071      3073      3075      3077      3079      3081      3083      3085      3087      3089      3091      3093      3095      3097      3099      3101      3103      3105      3107      3109      3111      3113      3115      3117      3119      3121      3123      3125      3127      3129      3131      3133      3135      3137      3139      3141      3143      3145      3147      3149      3151      3153      3155      3157      3159      3161      3163      3165      3167      3169      3171      3173      3175      3177      3179      3181      3183      3185      3187      3189      3191      3193      3195      3197      3199      3201      3203      3205      3207      3209      3211      3213      3215      3217      3219      3221      3223      3225      3227      3229      3231      3233      3235      3237      3239      3241      3243      3245      3247      3249      3251      3253      3255      3257      3259      3261      3263      3265      3267      3269      3271      3273      3275      3277      3279      3281      3283      3285      3287      3289      3291      3293      3295      3297      3299      3301      3303      3305      3307      3309      3311      3313      3315      3317      3319      3321      3323      3325      3327      3329      3331      3333      3335      3337      3339      3341      3343      3345      3347      3349      3351      3353      3355      3357      3359      3361      3363      3365      3367      3369      3371      3373      3375      3377      3379      3381      3383      3385      3387      3389      3391      3393      3395      3397      3399      3401      3403      3405      3407      3409      3411      3413      3415      3417      3419      3421      3423      3425      3427      3429      3431      3433      3435      3437      3439      3441      3443      3445      3447      3449      3451      3453      3455      3457      3459      3461      3463      3465      3467      3469      3471      3473      3475      3477      3479      3481      3483      3485      3487      3489      3491      3493      3495      3497      3499      3501      3503      3505      3507      3509      3511      3513      3515      3517      3519      3521      3523      3525      3527      3529      3531      3533      3535      3537      3539      3541      3543      3545      3547      3549      3551      3553      3555      3557      3559      3561      3563      3565      3567      3569      3571      3573      3575      3577      3579      3581      3583      3585      3587      3589      3591      3593      3595      3597      3599      3601      3603      3605      3607      3609      3611      3613      3615      3617      3619      3621      3623      3625      3627      3629      3631      3633      3635      3637      3639      3641      3643      3645      3647      3649      3651      3653      3655      3657      3659      3661      3663      3665      3667      3669      3671      3673      3675      3677      3679      3681      3683      3685      3687      3689      3691      3693      3695      3697      3699      3701      3703      3705      3707      3709      3711      3713      3715      3717      3719      3721      3723      3725      3727      3729      3731      3733      3735      3737      3739      3741      3743      3745      3747      3749      3751      3753      3755      3757      3759      3761      3763      3765      3767      3769      3771      3773      3775      3777      3779      3781      3783      3785      3787      3789      3791      3793      3795      3797      3799      3801      3803      3805      3807      3809      3811      3813      3815      3817      3819      3821      3823      3825      3827      3829      3831      3833      3835      3837      3839      3841      3843      3845      3847      3849      3851      3853      3855      3857      3859      3861      3863      3865      3867      3869      3871      3873      3875      3877      3879      3881      3883      3885      3887      3889      3891      3893      3895      3897      3899      3901      3903      3905      3907      3909      3911      3913      3915      3917      3919      3921      3923      3925      3927      3929      3931      3933      3935      3937      3939      3941      3943      3945      3947      3949      3951      3953      3955      3957      3959      3961      3963      3965      3967      3969      3971      3973      3975      3977      3979      3981      3983      3985      3987      3989      3991      3993      3995      3997      3999      4001      4003      4005      4007      4009      4011      4013      4015      4017      4019      4021      4023      4025      4027      4029      4031      4033      4035      4037      4039      4041      4043      4045      4047      4049      4051      4053      4055      4057      4059      4061      4063      4065      4067      4069      4071      4073      4075      4077      4079      4081      4083      4085      4087      4089      4091      4093      4095      4097      4099      4101      4103      4105      4107      4109      4111      4113      4115      4117      4119      4121      4123      4125      4127      4129      4131      4133      4135      4137      4139      4141      4143      4145      4147      4149      4151      4153      4155      4157      4159      4161      4163      4165      4167      4169      4171      4173      4175      4177      4179      4181      4183      4185      4187      4189      4191      4193      4195      4197      4199      4201      4203      4205      4207      4209      4211      4213      4215      4217      4219      4221      4223      4225      4227      4229      4231      4233      4235      4237      4239      4241      4243      4245      4247      4249      4251      4253      4255      4257      4259      4261      4263      4265      4267      4269      4271      4273      4275      4277      4279      4281      4283      4285      4287      4289      4291      4293      4295      4297      4299      4301      4303      4305      4307      4309
```

Output

```
X java
Sent: 120 2016-08-16 15:54:03 0 0 -97.83058333333334 30.44643333333335
Sent: 120 2016-08-16 15:54:12 0 0 -97.83058333333334 30.44643333333335
Sent: 120 2016-08-16 15:54:22 0 0 -97.83058333333334 30.44643333333335
Sent: 120 2016-08-16 15:54:32 0 0 -97.83058333333334 30.44643333333335
Sent: 120 2016-08-16 15:54:42 0 0 -97.83058333333334 30.44643333333335
Sent: 120 2016-08-16 15:54:52 0 0 -97.83058333333334 30.44643333333335
Sent: 120 2016-08-16 15:55:02 0 0 -97.83058333333334 30.44643333333335
Sent: 120 2016-08-16 15:55:12 0 0 -97.83058333333334 30.44643333333335
Sent: 113 2016-08-16 15:55:52 3.5 // 325 -97.83055833333333 30.446461666666668
Sent: 120 2016-08-16 15:55:56 0 0 -97.83058333333334 30.44643333333335
Sent: 107 2016-08-16 15:55:58 0 0 -97.83049166666666 30.446375
Sent: 120 2016-08-16 15:56:06 0 // let's hook up the busines 30.44643333333335
ties
Sent: 120 2016-08-16 15:56:16 0 KafkaStreams s 30.44643333333335
64 // and then,, we can start the stream processing
```

Build and Start the Processor

```
x sh
LM-SJN-21001415:processor pgraff$ mvn package
[INFO] Scanning for projects...
[INFO]
[INFO] -----
[INFO] Building iot-example 1.0-SNAPSHOT
[INFO]
```

```
x sh
[INFO]
[INFO] --- really-executable-jar-maven-plugin:1.1.0:really-executable-jar (default) @ iot-example ---
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 4.037 s
[INFO] Finished at: 2017-11-12T11:55:59-06:00
[INFO] Final Memory: 50M/368M
[INFO] -----
LM-SJN-21001415:processor pgraff$ target/parking-processor
```

Output

```
java
Calculating geohash for 88(-97.4026666666666,27.765366666666665) limit=1
Calculating geohash for 88(-97.4026666666666,27.765366666666665) lue>
Calculating geohash for 88(-97.4026666666666,27.76546666666667) ere>
Calculating geohash for 88(-97.4026666666666,27.76546666666667) length=
Calculating geohash for 22(-97.8934666666667,30.369886666666666)
Calculating geohash for 22(-97.8934666666667,30.36988333333334)
Calculating geohash for 22(-97.9058,30.37845)
Calculating geohash for 22(-97.9058,30.378416666666666)
Calculating geohash for 88(-97.40265,27.7654333333334)
Calculating geohash for 88(-97.4026633333334,27.76544666666666)
Calculating geohash for 88(-97.4026333333333,27.7654666666667)
Calculating geohash for 88(-97.4026483333333,27.765475)
Calculating geohash for 88(-97.4027,27.7654)
Calculating geohash for 88(-97.402715,27.76541)
Calculating geohash for 88(-97.4026666666666,27.76545)
Calculating geohash for 88(-97.4026733333334,27.765455);
Calculating geohash for 88(-97.4026733333334,27.765455)
```

docker-compose	
120@9v6m	5934
111@9v6m	1035
107@9v6m	2208
13@9v6m	1575
22@9v6m	1903
22@9v6k	4588
88@9v6m	2464
22@9v6e	146
88@9uft	4587
114@9v6m	144
110@9v6m	144
88@9ufq	648
113@9v6m	2016
88@9ufw	360
120@9v6m	6192
111@9v6m	1080
107@9v6m	2304
13@9v6m	1638
22@9v6m	1981
22@9v6k	4773

Data is produced
in the message-
pump

Then Processed in
the parking-
processor

Then finally
printed in the
console consumer