Stream Processing: Using Kafka and KSQL for Twitter data

By: Allen Ansari, Yonjun “Ian” Chu, Chad Madding, Emil Ramos

Agenda

- Problem statement
- Project Overview
- Kafka
- KSQL
- Stream Processing with Twitter data

Problem Statement

With today’s ever-increasing demand for real-time analytics, traditional batch-oriented data processing doesn’t suffice.

Approach:

- Identify how stream processing allows us to harness the full value of the data; extracting valuable information.
- Use stream processing to automatically filter, aggregate, and analyze Twitter data

Project Overview

Read tweets from Twitter's API by using KSQL's ability to look at Kafka event streams

Implement continuous stream processing from Python environments by writing simple SQL-like statements

Identify how Kafka and KSQL will help us analyze Twitter data and implement desirable machine learning methods to understand underlying information in Twitter data.

Stream Processing

Stream Processing allows users to harness the full value of data as it is produced; extracting valuable information in motion

Requires different tools from those used in traditional batch processing architecture. We will use Apache **Kafka** and **KSQL**

Kafka takes data published by ‘producers’, which maybe apps, files, or databases, and makes it available for ‘consumers’ subscribed to streams of different ‘topics.’

KSQL, built on top of Kafka’s Streams API, supports stream processing operations like filtering, transformations, aggregations, joins and windowing by using SQL statements instead of writing a lot of codes

KAFKA capabilities

- Distributed Streaming Platform that can publish and subscribe to streams of records
- Fault tolerant storage
- Replicates Topic Log Partitions to multiple servers
- Process records as they occur
- Fast, efficient IO, batching, compression, and more
- Used to decouple data streams

Kafka Use Cases

- Stream Processing
- Website Activity Tracking
- Metrics Collection and Monitoring
- Log Aggregation
- Real time analytics
- Capture and ingest data into Spark / Hadoop
- CRQS, replay, error recovery
- Guaranteed distributed commit log for in-memory computing

Kafka Fundamentals

Records have a key (optional), value and timestamp; Immutable

Topic a stream of records (“/orders”, “/user-signups”), feed name

Log topic storage on disk

Partition / Segments (parts of Topic Log)

Producer API to produce a streams or records

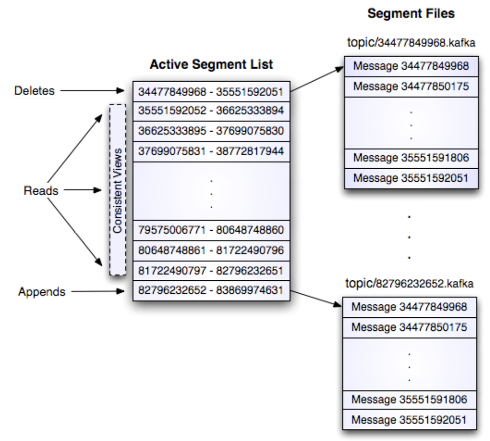
Consumer API to consume a stream of records

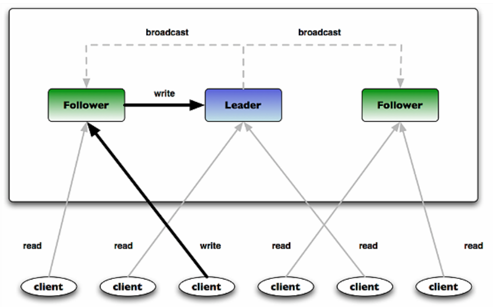
Broker: Kafka server that runs in a Kafka Cluster. Brokers form a cluster. Cluster consists on many Kafka Brokers on many servers.

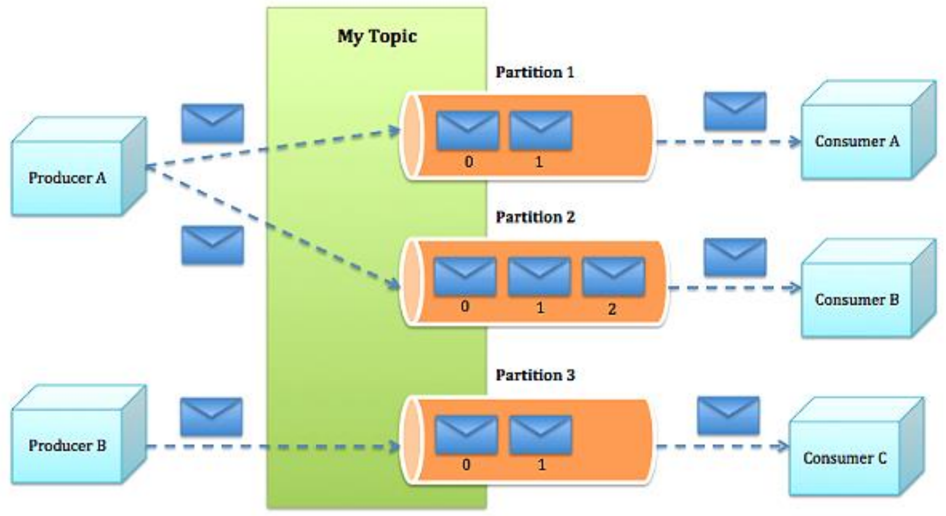
ZooKeeper: Does coordination of brokers/cluster topology. Consistent file system for configuration information and leadership election for Broker Topic Partition Leaders

KAFKA Architecture

- Each partition of a topic corresponds to a logical log
- Physically, a log is implemented as a set of segment files of equal sizes
- Every time a producer publishes a message to a partition, the broker simply appends the message to the last segment file
- Segment file is flushed to disk after configurable numbers of messages have been published or after a certain amount of time elapsed
- Messages are exposed to consumer after it gets flushed.
- Consumer always consumes messages from a particular partition sequentially and if the consumer acknowledges particular message offset, it implies that the consumer has consumed all prior messages.





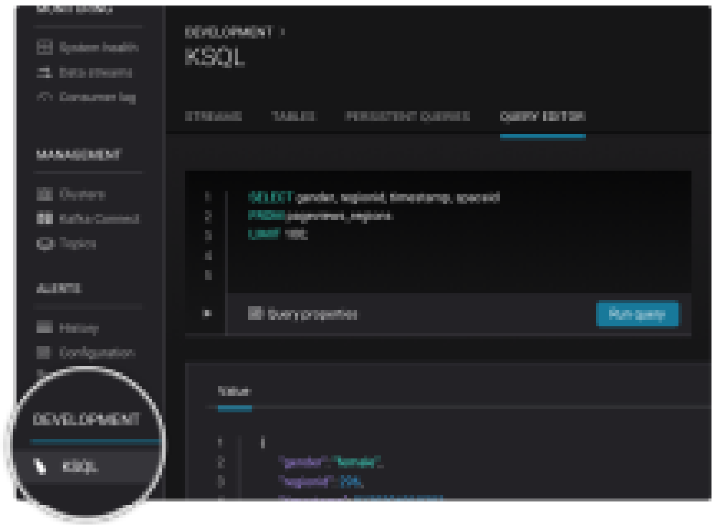


KSQL

- Open Source
- Zero Programming in Java, Scala
- Elastic, Scalable, Fault-Tolerant, Distributed
- Powerful Processing incl. Filters, Transforms, Joins, Aggregations, Windowing
- Runs Everywhere
- Supports Streams and Tables
- Exactly-Once Processing
- Event-Time Processing
- Kafka Security Integration

Interactive KSQL usage

```
ksql>
```



```
SELECT gender, regionid, timestamp, spaceid
FROM pageviews_regions
LIMIT 100;
```

```
{
    "gender": "female",
    "regionid": 206,
```

```
POST /query
```

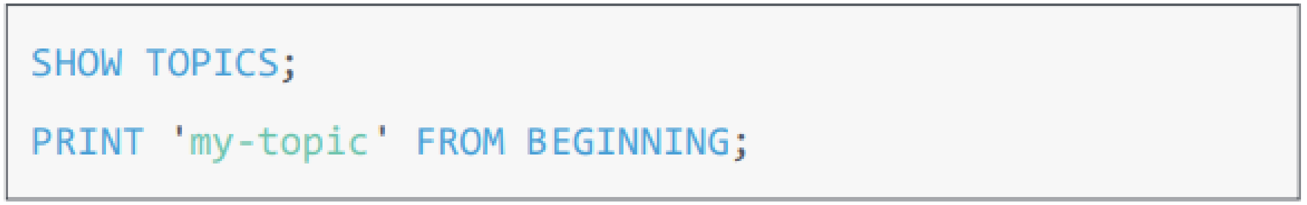
1 CLI

2 UI

3 REST API

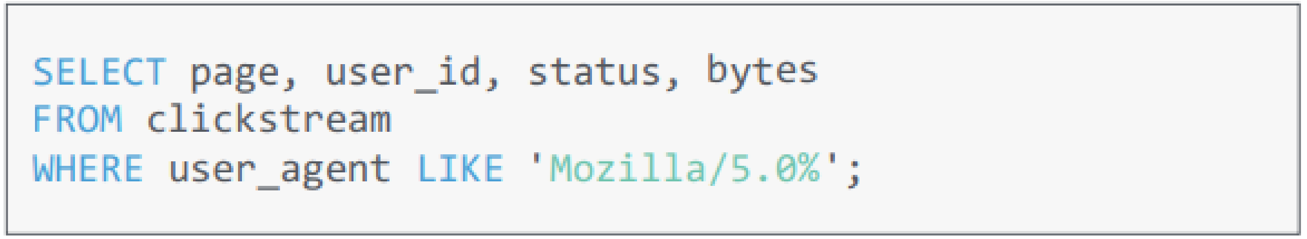
KSQL for Data Exploration

- An easy way to inspect data in Kafka
- Join data from a variety of sources to see the full picture

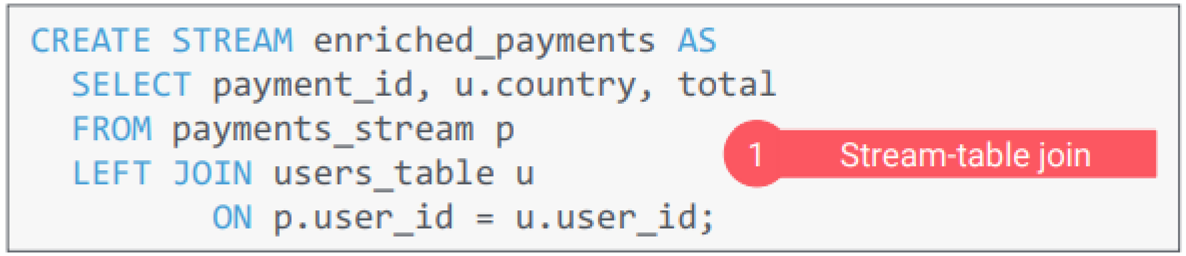


```
SHOW TOPICS;

PRINT 'my-topic' FROM BEGINNING;
```



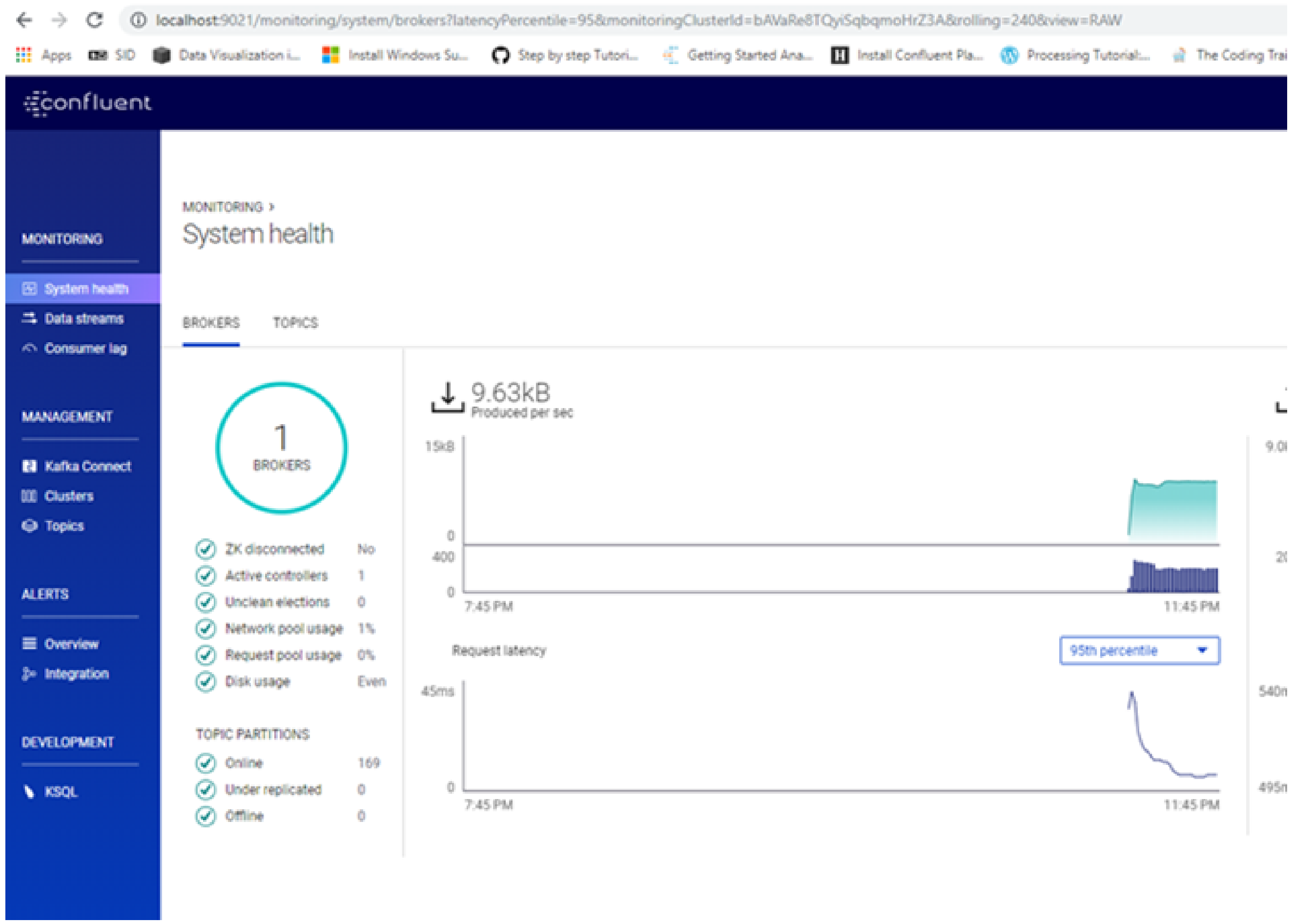
```
SELECT page, user_id, status, bytes
FROM clickstream
WHERE user_agent LIKE 'Mozilla/5.0%';
```



```
CREATE STREAM enriched_payments AS
  SELECT payment_id, u.country, total
  FROM payments_stream p
  LEFT JOIN users_table u
         ON p.user_id = u.user_id;
```

Software Setup Confluent Kafka

- Installed Confluent Kafka on pure Linux environment
  - Issue with the Windows 10 Linux install
- Ubuntu 18 was chosen because of its ease of installation and graphical environment
  - Ubuntu 18 is a open source Debian-based Linux distribution

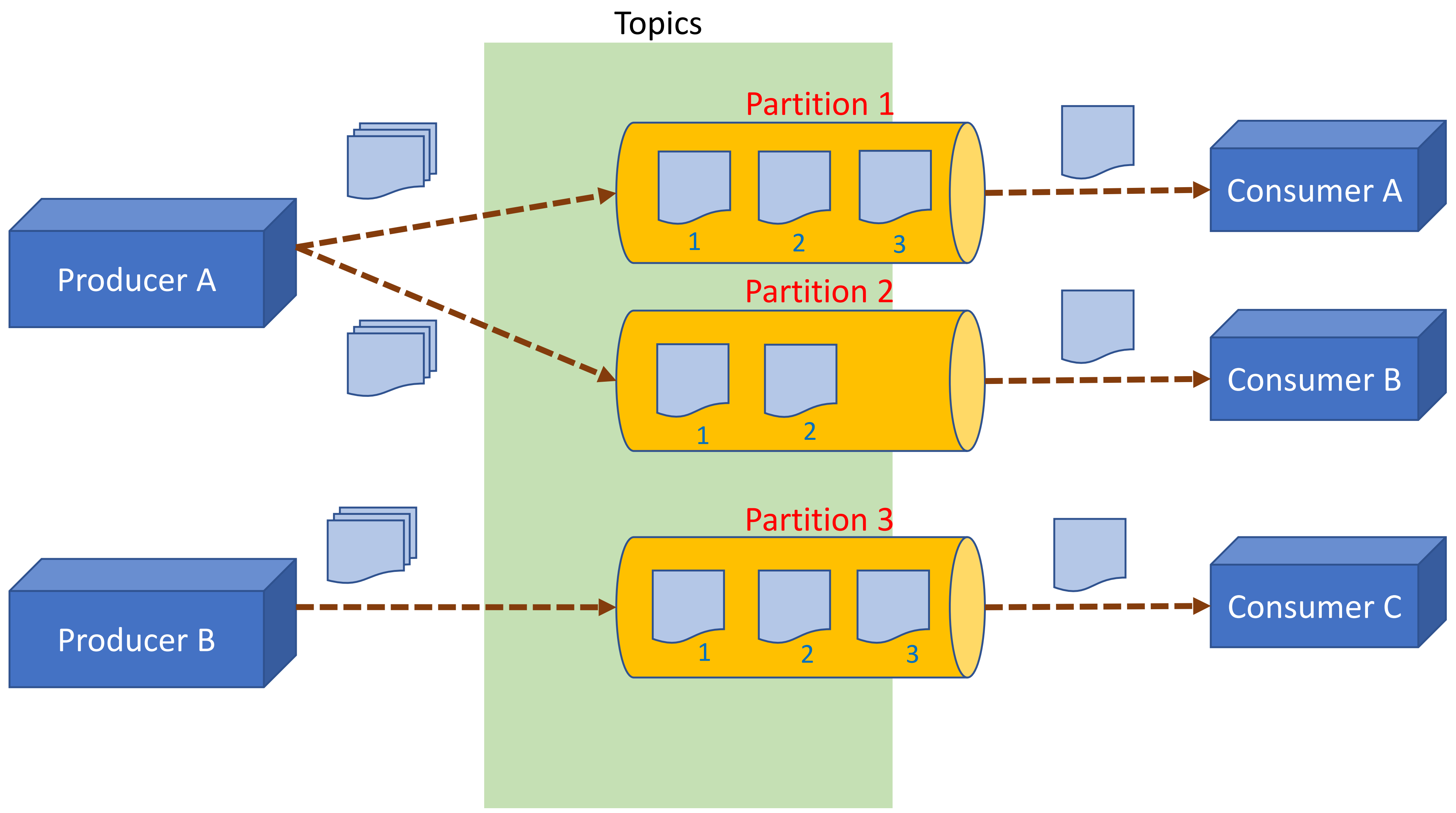


ZK disconnected	No
Active controllers	1
Unclean elections	0
Network pool usage	1%
Request pool usage	0%
Disk usage	Even

TOPIC PARTITIONS

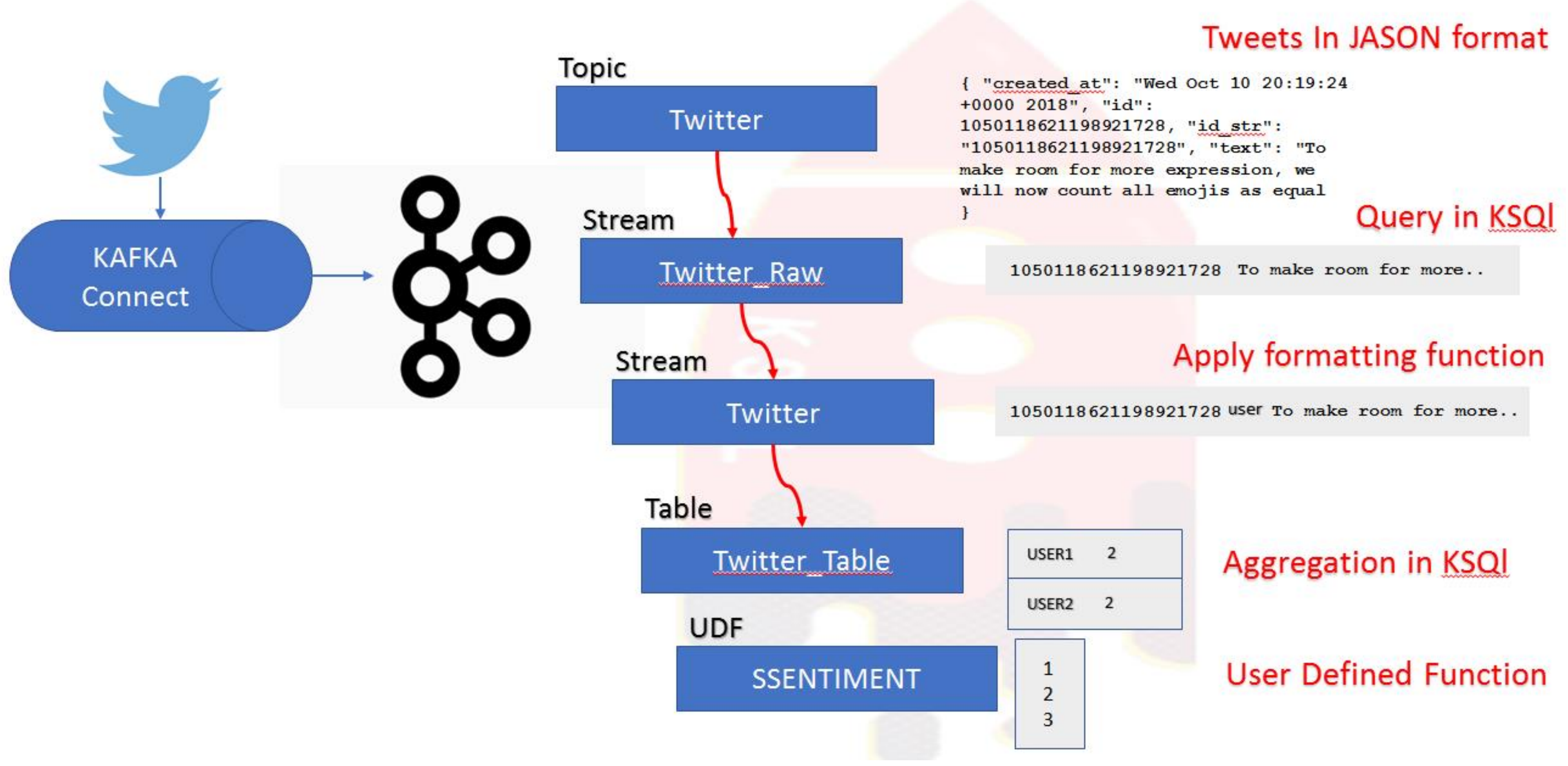
Online	169
Under replicated	0
Offline	0

KSQL is the querying side of Kafka. It can be ran in both command line and a graphical environment and run from a web page on the local computer


Topics
Partition 1
1
2
3
Partition 2
1
2
Partition 3
1
2
3
Producer A
Producer B
Consumer A
Consumer B
Consumer C

Analyzing Twitter Data in Apache Kafka through KSQL

- Used Kafka Connect to pull the data from Twitter
- Used Twitter connector that uses the twitter streaming API to listen for status update messages and convert them to a Kafka Connect struct on the fly.
    - The goal is to match as much of the Twitter Status object as possible



```
{ "created_at": "Wed Oct 10 20:19:24
+0000 2018", "id":
1050118621198921728, "id_str":
"1050118621198921728", "text": "To
make room for more expression, we
will now count all emojis as equal
}
```

```
1050118621198921728  To make room for more..
```

```
1050118621198921728 user To make room for more..
```

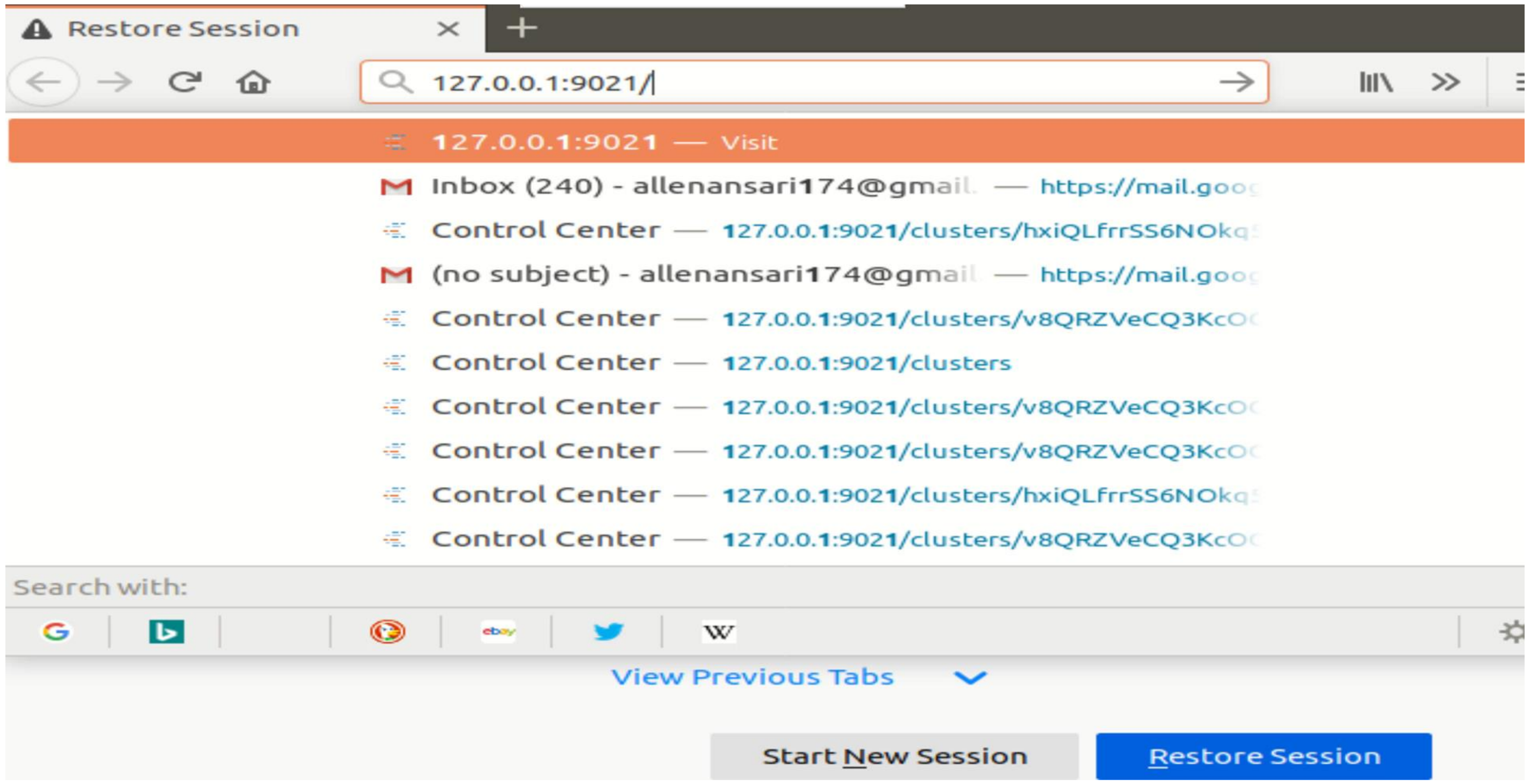
USER1	2
USER2	2

1
2
3

KSQL CLI

```
allen@allen:~$ 
```

KSQL UI


Restore Session

127.0.0.1:9021/

127.0.0.1:9021 — Visit

Inbox (240) - allenansari174@gmail — https://mail.goo

Control Center — 127.0.0.1:9021/clusters/hxiQLfrrSS6NOkq

(no subject) - allenansari174@gmail — https://mail.goo

Control Center — 127.0.0.1:9021/clusters/v8QRZVeCQ3KcO

Control Center — 127.0.0.1:9021/clusters

Control Center — 127.0.0.1:9021/clusters/v8QRZVeCQ3KcO

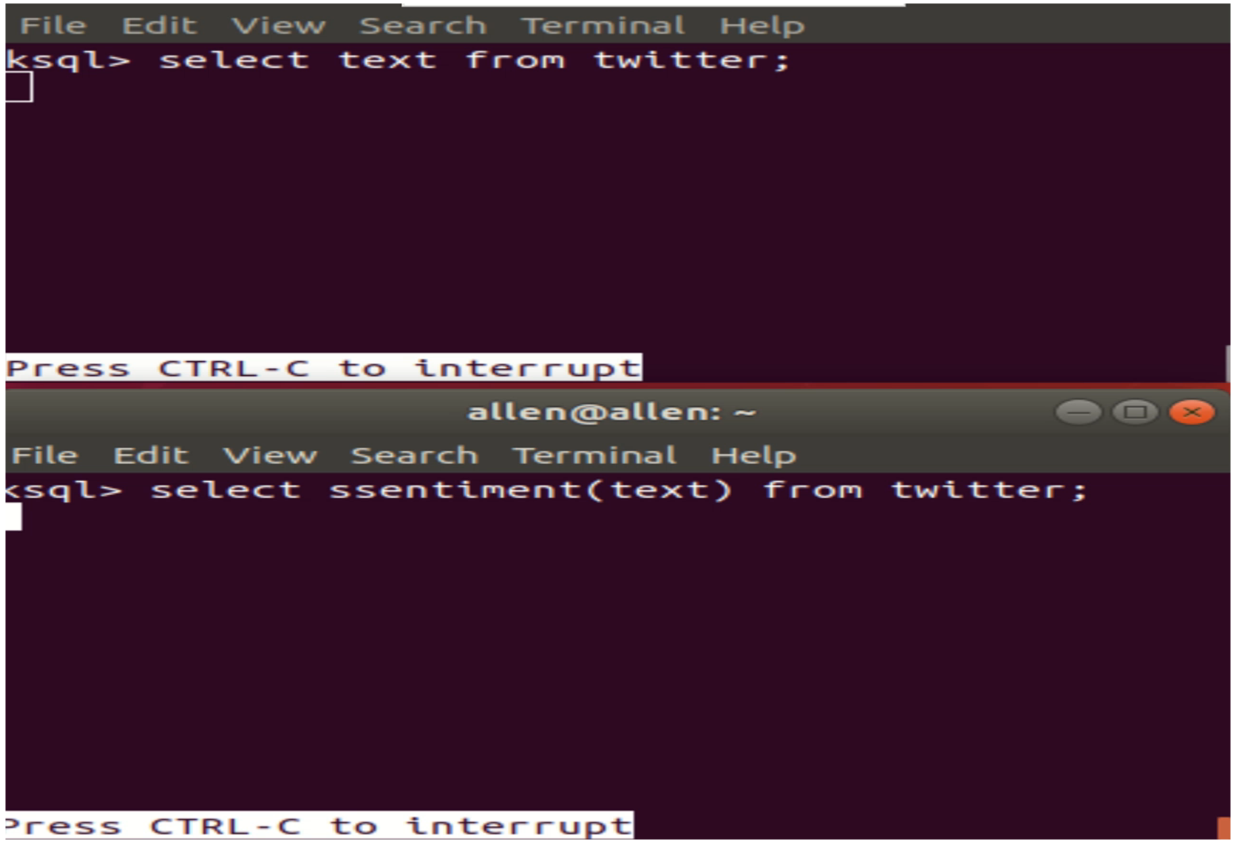
Control Center — 127.0.0.1:9021/clusters/v8QRZVeCQ3KcO

Control Center — 127.0.0.1:9021/clusters/hxiQLfrrSS6NOkq

Control Center — 127.0.0.1:9021/clusters/v8QRZVeCQ3KcO

Search with:

Twitter Sentiment Analysis


File Edit View Search Terminal Help

```
ksql> select text from twitter;

Press CTRL-C to interrupt
```

allen@allen: ~
File Edit View Search Terminal Help

```
ksql> select ssentiment(text) from twitter;

Press CTRL-C to interrupt
```

1 = Negative
2 = Neutral
3 = Positive