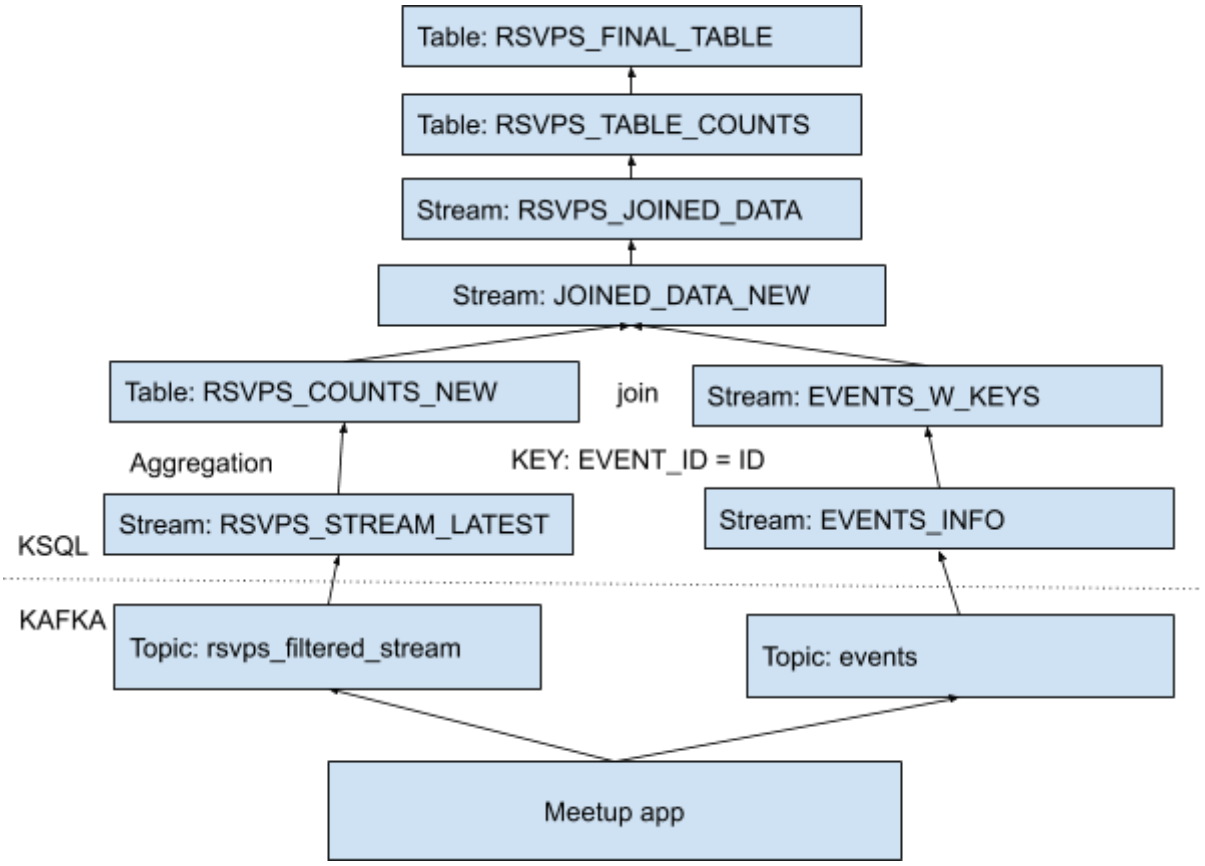
# Kafka - Detailed Architecture



## RSVPs Distribution




**Data that is written in a Kafka topic does not have any schema. This means that we cannot do common operations such as aggregation or filtering on it. For this reason we define a stream that structures the data from the topic.**

CREATE STREAM **RSVPS_STREAM_LATEST** \

```
    (
      RSVP_ID BIGINT, \
      VENUE STRUCT<VENUE_NAME VARCHAR, LON DOUBLE, LAT DOUBLE, \
VENUE_ID BIGINT>,
      MEMBER_ID BIGINT, \
      MTIME BIGINT, \
      RESPONSE VARCHAR, \
      GUESTS BIGINT, \
      EVENT_ID VARCHAR, \
      GROUP_URLNAME VARCHAR, \
      GROUP_TOPICS ARRAY <STRUCT<TOPIC_NAME VARCHAR, URLKEY
VARCHAR>>) \
WITH (KAFKA_TOPIC='rsvps_filtered_stream',
VALUE_FORMAT='JSON',TIMESTAMP='mtime');
```

**We need a way to sum the count of the real time RSVPs from all events. To do this we create a Table with the RSVP counts per each event.**

```
SET 'auto.offset.reset'='earliest'; // To force reading the stream from the start

CREATE TABLE RSVPS_COUNTS_NEW WITH
(KAFKA_TOPIC='RSVPS_COUNTS_NEW', PARTITIONS=1, REPLICAS=1) AS SELECT
  RSVPS_STREAM_LATEST.EVENT_ID "EVENT_ID",
  COLLECT_SET(RSVPS_STREAM_LATEST.VENUE->VENUE_NAME) "VENUE_NAME",
  COLLECT_SET(RSVPS_STREAM_LATEST.VENUE->LAT) "VENUE_LAT",
  COLLECT_SET(RSVPS_STREAM_LATEST.VENUE->LON) "VENUE_LON",
  COUNT(*) "NUMBER_RSVPS",
  SUM(RSVPS_STREAM_LATEST.GUESTS) "GUESTS_COUNT"
FROM RSVPS_STREAM_LATEST RSVPS_STREAM_LATEST
WHERE (RSVPS_STREAM_LATEST.RESPONSE = 'yes')
GROUP BY RSVPS_STREAM_LATEST.EVENT_ID
EMIT CHANGES;
```

**Again we need to create another stream, this time over the events topic.**

```
CREATE STREAM EVENTS_INFO \
(created BIGINT, \
duration BIGINT, \
id VARCHAR, \
name VARCHAR, \
status VARCHAR, \
time BIGINT, \
waitlist_count BIGINT, \
yes_rsvp_count BIGINT, \
```

```
venue STRUCT< \
        id BIGINT, \
        name VARCHAR, \
        repinned VARCHAR, \
        country VARCHAR, \
        localized_country_name varchar>, \
link VARCHAR, \
description VARCHAR, \
event_group STRUCT< \
        name VARCHAR, \
        urlname VARCHAR, \
        lon DOUBLE, \
        lat DOUBLE, \
        localized_location VARCHAR, \
        state VARCHAR, \
        country VARCHAR, \
        region VARCHAR, \
        timezone VARCHAR> \
) \
WITH (KAFKA_TOPIC='events', VALUE_FORMAT='JSON');
```

**As you can see some of the data comes in as a struct type. Also we do not need all of the fields. Therefore we create another stream to keep only what's relevant only with simple columns**

```
CREATE STREAM EVENTS_W_KEYS WITH (KAFKA_TOPIC='events_w_keys',
PARTITIONS=1, REPLICAS=1) AS SELECT
  EVENTS_INFO.CREATED "CREATED",
  EVENTS_INFO.ID "ID",
  EVENTS_INFO.NAME "NAME",
  EVENTS_INFO.WAITLIST_COUNT "WAITLIST_COUNT",
  EVENTS_INFO.YES_RSVP_COUNT "YES_RSVP_COUNT",
  EVENTS_INFO.EVENT_GROUP->URLNAME "EVENT_GROUP__URLNAME",
  EVENTS_INFO.EVENT_GROUP->LON "EVENT_GROUP__LON",
  EVENTS_INFO.EVENT_GROUP->LAT "EVENT_GROUP__LAT",
  EVENTS_INFO.EVENT_GROUP->LOCALIZED_LOCATION
"EVENT_GROUP__LOCALIZED_LOCATION",
  EVENTS_INFO.EVENT_GROUP->STATE "EVENT_GROUP__STATE",
  EVENTS_INFO.EVENT_GROUP->COUNTRY "EVENT_GROUP__COUNTRY",
  EVENTS_INFO.EVENT_GROUP->REGION "EVENT_GROUP__REGION"
FROM EVENTS_INFO EVENTS_INFO
EMIT CHANGES
PARTITION BY ID;
```

**Now, we have a stream with Event_ID as key - including the event detailed info. Let's join events data with RSVP data.**

```
CREATE STREAM JOINED_DATA_NEW WITH (KAFKA_TOPIC='JOINED_DATA_NEW',
PARTITIONS=1, REPLICAS=1) AS SELECT
  EVENT.ID "EVENTID",
  EVENT.NAME "NAME",
  EVENT.WAITLIST_COUNT "WAITLIST_COUNT",
  EVENT.YES_RSVP_COUNT "YES_RSVP_COUNT",
  EVENT.EVENT_GROUP__URLNAME "EVENT_GROUP__URLNAME",
  EVENT.EVENT_GROUP__LON "EVENT_GROUP__LON",
  EVENT.EVENT_GROUP__LAT "EVENT_GROUP__LAT",
  EVENT.EVENT_GROUP__LOCALIZED_LOCATION
"EVENT_GROUP__LOCALIZED_LOCATION",
  EVENT.EVENT_GROUP__STATE "EVENT_GROUP__STATE",
  EVENT.EVENT_GROUP__COUNTRY "EVENT_GROUP__COUNTRY",
  EVENT.EVENT_GROUP__REGION "EVENT_GROUP__REGION",
  C.VENUE_NAME "VENUE_NAME",
  C.VENUE_LAT "VENUE_LAT",
  C.VENUE_LON "VENUE_LON",
  C.NUMBER_RSVPS "NUMBER_RSVPS",
  C.GUESTS_COUNT "GUESTS_COUNT"
FROM EVENTS_W_KEYS EVENT
INNER JOIN RSVPS_COUNTS_NEW C ON ((EVENT.ID = C.EVENT_ID))
EMIT CHANGES;
```

**We now want to find out all RSVPs (initial, counted since and guests) for an event id.**

```
CREATE STREAM RSVPS_JOINED_DATA WITH
(KAFKA_TOPIC='RSVPS_JOINED_DATA', PARTITIONS=1, REPLICAS=1) AS SELECT
  JOINED_DATA_NEW.EVENTID "EVENTID",
  JOINED_DATA_NEW.NAME "NAME",
  JOINED_DATA_NEW.EVENT_GROUP__URLNAME "EVENT_GROUP__URLNAME",
  JOINED_DATA_NEW.EVENT_GROUP__LON "EVENT_GROUP__LON",
  JOINED_DATA_NEW.EVENT_GROUP__LAT "EVENT_GROUP__LAT",
  JOINED_DATA_NEW.EVENT_GROUP__LOCALIZED_LOCATION
"EVENT_GROUP__LOCALIZED_LOCATION",
  JOINED_DATA_NEW.EVENT_GROUP__STATE "EVENT_GROUP__STATE",
  JOINED_DATA_NEW.EVENT_GROUP__COUNTRY "EVENT_GROUP__COUNTRY",
  JOINED_DATA_NEW.EVENT_GROUP__REGION "EVENT_GROUP__REGION",
  JOINED_DATA_NEW.VENUE_NAME "VENUE_NAME",
  JOINED_DATA_NEW.VENUE_LAT "VENUE_LAT",
  JOINED_DATA_NEW.VENUE_LON "VENUE_LON",
```

```
  (((JOINED_DATA_NEW.WAITLIST_COUNT +
JOINED_DATA_NEW.YES_RSVP_COUNT) + JOINED_DATA_NEW.NUMBER_RSVPS) +
JOINED_DATA_NEW.GUESTS_COUNT) "RSVPS_TOTAL"
FROM JOINED_DATA_NEW JOINED_DATA_NEW
EMIT CHANGES;
```

**We create a table with sum all of RSVPs.**

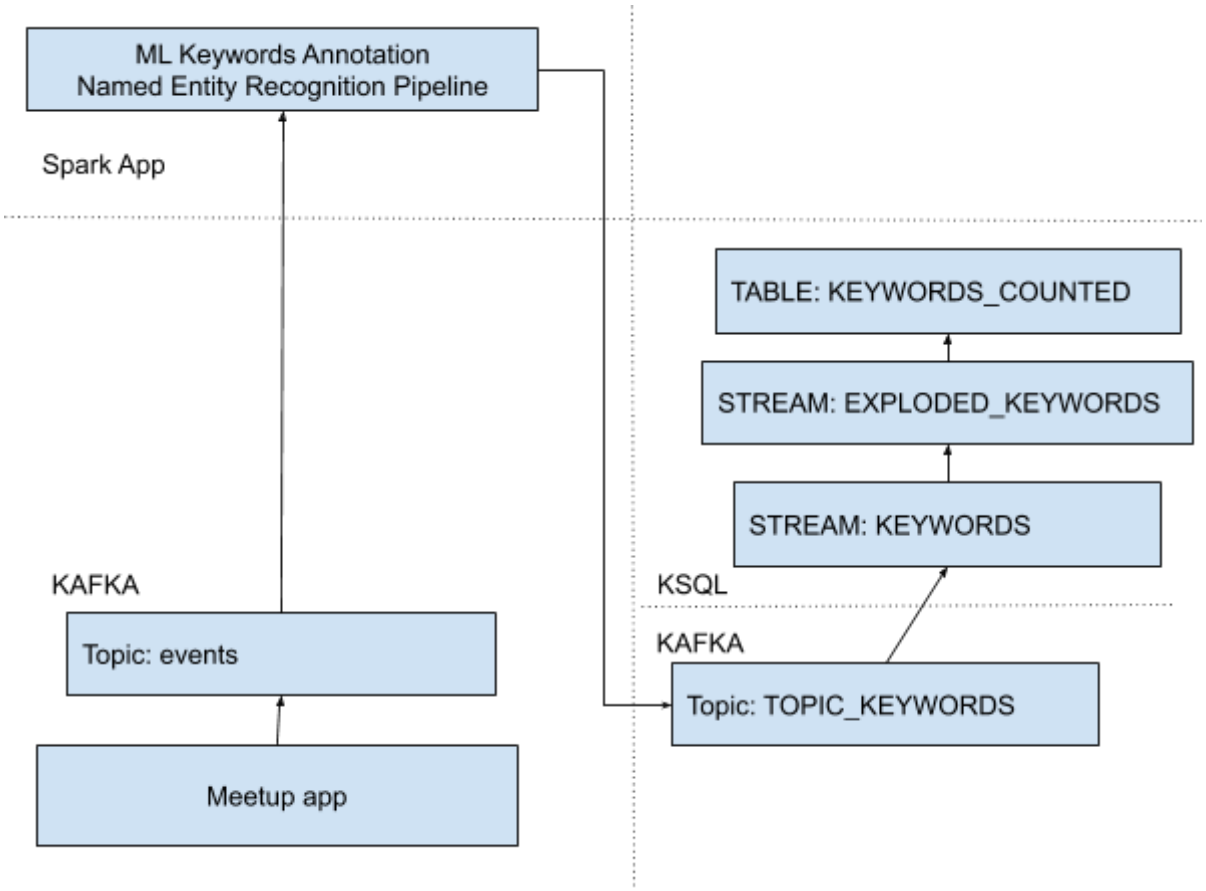
```
CREATE TABLE RSVPS_TABLE_COUNTS WITH
(KAFKA_TOPIC='RSVPS_TABLE_COUNTS', PARTITIONS=1, REPLICAS=1) AS SELECT
  RSVPS_JOINED_DATA.EVENTID "EVENTID",
  COLLECT_SET(RSVPS_JOINED_DATA.NAME) "NAME",
  COLLECT_SET(RSVPS_JOINED_DATA.EVENT_GROUP__URLNAME)
"EVENT_GROUP__URLNAME",
  COLLECT_SET(RSVPS_JOINED_DATA.EVENT_GROUP__LON)
"EVENT_GROUP_LON",
  COLLECT_SET(RSVPS_JOINED_DATA.EVENT_GROUP__LAT)
"EVENT_GROUP__LAT",
  COLLECT_SET(RSVPS_JOINED_DATA.EVENT_GROUP__LOCALIZED_LOCATION)
"EVENT_GROUP__LOCALIZED_LOCATION",
  COLLECT_SET(RSVPS_JOINED_DATA.EVENT_GROUP__STATE)
"EVENT_GROUP__STATE",
  COLLECT_SET(RSVPS_JOINED_DATA.EVENT_GROUP__COUNTRY)
"EVENT_GROUP__COUNTRY",
  COLLECT_SET(RSVPS_JOINED_DATA.EVENT_GROUP__REGION)
"EVENT_GROUP__REGION",
  MAX(RSVPS_JOINED_DATA.RSVPS_TOTAL) "RSVPS_COUNT"
FROM RSVPS_JOINED_DATA RSVPS_JOINED_DATA
GROUP BY RSVPS_JOINED_DATA.EVENTID
EMIT CHANGES;
```

**Finally we create a table that contains only the last count.**

```
CREATE TABLE RSVPS_FINAL_TABLE WITH (KAFKA_TOPIC='RSVPS_FINAL_TABLE',
PARTITIONS=1, REPLICAS=1) AS SELECT
  RSVPS_TABLE_COUNTS.EVENTID "EVENTID",
  CAST(RSVPS_TABLE_COUNTS.NAME AS STRING) "NAME",
  CAST(RSVPS_TABLE_COUNTS.EVENT_GROUP__URLNAME AS STRING)
"EVENT_GROUP__URLNAME",
  CAST(RSVPS_TABLE_COUNTS.EVENT_GROUP_LON AS STRING)
"EVENT_GROUP_LON",
  CAST(RSVPS_TABLE_COUNTS.EVENT_GROUP__LAT AS STRING)
"EVENT_GROUP__LAT",
```

```
  CAST(RSVPS_TABLE_COUNTS.EVENT_GROUP__LOCALIZED_LOCATION AS
STRING) "EVENT_GROUP__LOCALIZED_LOCATION",
  CAST(RSVPS_TABLE_COUNTS.EVENT_GROUP__STATE AS STRING)
"EVENT_GROUP__STATE",
  CAST(RSVPS_TABLE_COUNTS.EVENT_GROUP__COUNTRY AS STRING)
"EVENT_GROUP__COUNTRY",
  CAST(RSVPS_TABLE_COUNTS.EVENT_GROUP__REGION AS STRING)
"EVENT_GROUP__REGION",
  RSVPS_TABLE_COUNTS.RSVPS_COUNT "RSVPS_COUNT"
FROM RSVPS_TABLE_COUNTS RSVPS_TABLE_COUNTS
EMIT CHANGES;
```

## .Trending Keywords per Topic




**We create a stream that reads keywords from TOPIC_KEYWORDS which is created as a results of Spark NLP**

```
CREATE STREAM KEYWORDS \
    (
```

```
      event_id BIGINT, \
       event_timestamp BIGINT, \
       country VARCHAR, \
       keywords array <STRING>) \
WITH (KAFKA_TOPIC='TOPIC_KEYWORDS', VALUE_FORMAT='JSON');
```

**Keywords come per event in an array. In order to do aggregations on them we need to have each keyword on 1 line. Therefore we explode the keywords array.**

```
CREATE STREAM EXPLODED_KEYWORDS WITH
(KAFKA_TOPIC='EXPLODED_KEYWORDS', PARTITIONS=1, REPLICAS=1) AS SELECT
  KEYWORDS.EVENT_ID "EVENT_ID",
  KEYWORDS.EVENT_TIMESTAMP "EVENT_TIMESTAMP",
  KEYWORDS.COUNTRY "COUNTRY",
  EXPLODE(KEYWORDS.KEYWORDS) "KEYWORD"
FROM KEYWORDS KEYWORDS
EMIT CHANGES;
```

**Finally, we count the number of occurrences for each keyword, in all of the countries. We keep the countries where the keyword occurred in the event description as an array.**

```
CREATE TABLE KEYWORDS_COUNTED WITH
(KAFKA_TOPIC='KEYWORDS_COUNTED', PARTITIONS=1, REPLICAS=1) AS SELECT
  EXPLODED_KEYWORDS.KEYWORD "KEYWORD",
  COUNT(EXPLODED_KEYWORDS.KEYWORD) "COUNT_K",
  CAST(COLLECT_SET(EXPLODED_KEYWORDS.COUNTRY) AS STRING) "COUNTRY"
FROM EXPLODED_KEYWORDS EXPLODED_KEYWORDS
GROUP BY EXPLODED_KEYWORDS.KEYWORD
EMIT CHANGES;
```