In [1]:
```
sc
```

Out[1]:

SparkContext

Spark UI (http://172.16.86.148:4040)

Version

`v2.2.1`

Master

`local[2]`

AppName

`pyspark-shell`

Read the streams from kafka

In [2]:
```
df = spark                                                  \
  .readStream                                               \
  .format("kafka")                                          \
  .option("kafka.bootstrap.servers", "localhost:9092") \
  .option("subscribe", "tweets_topic")                 \
  .load()
```

Possible kafka read options

- Reading from multiple topics

```
option("subscribe", "topicA,topicB")
```

- Reading from topics with names following a pattern

```
option("subscribepattern", """topic\d""")
```

- Reading topics and partitions

```
assign({"topicA":[0,1],"topicB":[0,1]})
```

- Start reading from which offset

```
option("startingoffsets", "latest")  // or "earliest"
```

- How many reacord for each trigger

```
option("maxOffsetsPerTrigger", 1)
```

Extract tags and count them

- Information that spark reads for each record

```
val result = df.
  select(
    $"key" cast "string",   // deserialize keys
    $"value" cast "string", // deserialize values
    $"topic",
    $"partition",
    $"offset")
```

In [3]:
```
from pyspark.sql.functions import explode, split, col

words = df.select(explode(split(df.value, " ")).alias("token"))
tags = words.where(words.token.contains("#"))
tagcount = tags.groupBy("token").count()
```

Write results to the sink

In [4]:
```
query = tagcount.writeStream           \
         .outputMode("complete")       \
         .format("memory")             \
         .queryName("tweetstrends")    \
         .start()
```

- Setting trigger interval

```
trigger(Trigger.ProcessingTime(10.seconds))
```

- Checkpoint path to recover from failures

option("checkpointLocation", "path/to/HDFS/dir") \

Print the top 10 trends

```
In [9]: ## Show the top 10 tags
        spark.sql("Select token, count from tweetstrends order by count desc limit 10").show()
```

```
+------------------+-----+
|             token|count|
+------------------+-----+
|#INSTANTFOLLOWBACK|    1|
|         #handmade|    1|
|         #sidoarjo|    1|
|            #sweet|    1|
|              #art|    1|
|       #goodmoning|    1|
|        #colazione|    1|
|            #Aries|    1|
|            #final|    1|
|    #TEAMFOLLOWWACK|    1|
+------------------+-----+
```

Check Status

In [6]:
```
query.lastProgress
```

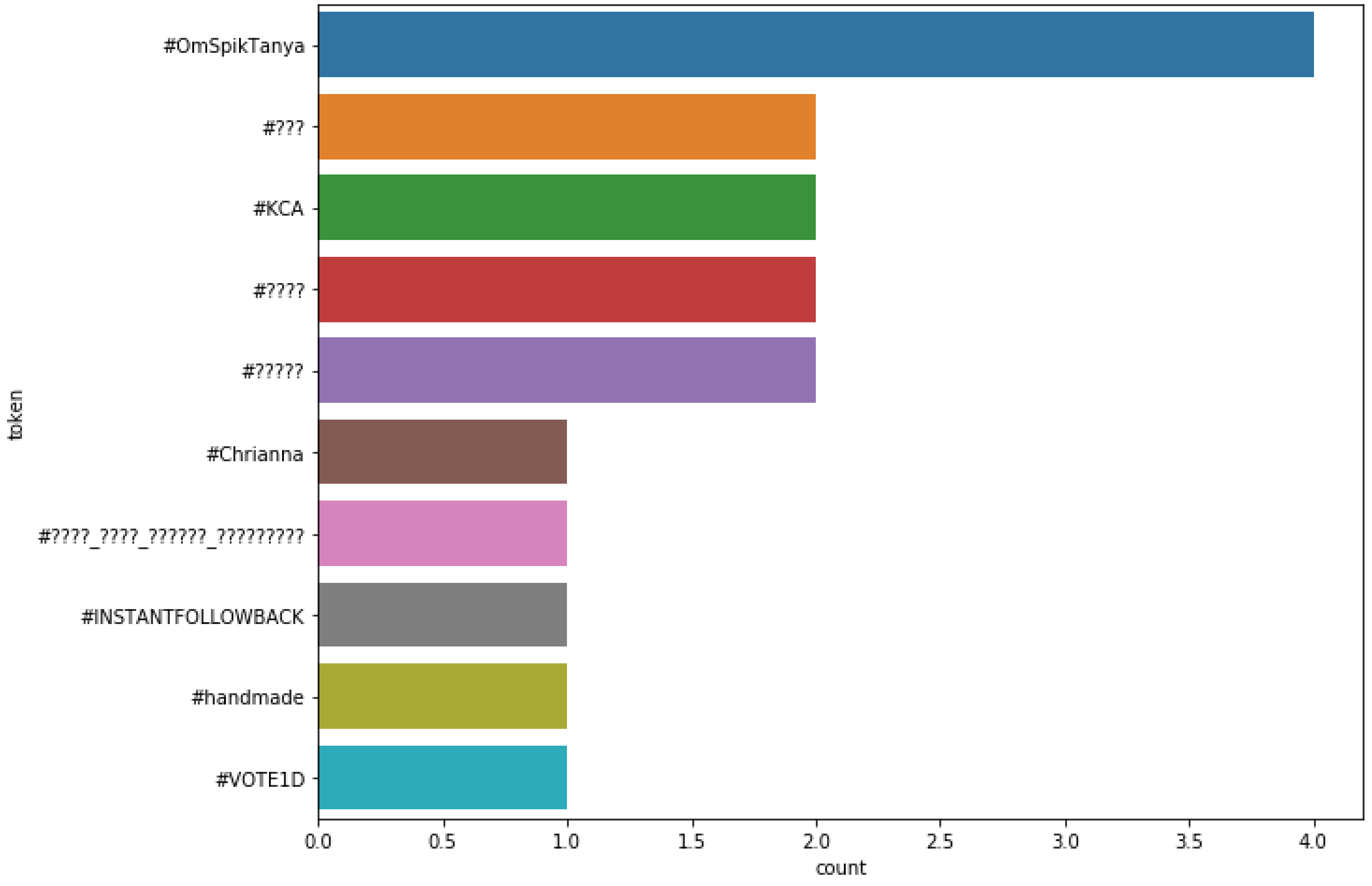
Out[6]:
```
{'id': 'b9d1ad9b-02c0-4f6f-817a-403e7acb2d32',
 'runId': 'da7147bb-3988-4a9e-b3d2-0a403b2275b7',
 'name': 'tweetstrends',
 'timestamp': '2018-08-28T20:14:19.624Z',
 'numInputRows': 23,
 'inputRowsPerSecond': 1533.3333333333335,
 'processedRowsPerSecond': 4.03013842649378,
 'durationMs': {'addBatch': 5588,
  'getBatch': 15,
  'getOffset': 1,
  'queryPlanning': 31,
  'triggerExecution': 5707,
  'walCommit': 68},
 'stateOperators': [{'numRowsTotal': 3, 'numRowsUpdated': 3}],
 'sources': [{'description': 'KafkaSource[Subscribe[tweets_topic]]',
   'startOffset': {'tweets_topic': {'0': 1219}},
   'endOffset': {'tweets_topic': {'0': 1242}},
   'numInputRows': 23,
   'inputRowsPerSecond': 1533.3333333333335,
   'processedRowsPerSecond': 4.03013842649378}],
 'sink': {'description': 'MemorySink'}}
```

Visualize the Trends

In [10]:
```
import matplotlib.pyplot as plt
import seaborn as sn
%matplotlib inline
```

In [11]:
```
import time
from IPython import display


count = 0
while count < 10:
  time.sleep( 1 )
  top_10_tweets = sqlContext.sql( 'Select token, count from tweetstrends order by count desc limit 10' )
  top_10_df = top_10_tweets.toPandas()
  display.clear_output(wait=True)
  plt.figure( figsize = ( 10, 8 ) )
  sn.barplot( x="count", y="token", data=top_10_df)
  plt.show()
  count = count + 1
```


#OmSpikTanya
#???
#KCA
#????
#?????
#Chrianna
#????_????_??????_?????????
#INSTANTFOLLOWBACK
#handmade
#VOTE1D
token
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
count

```
In [ ]:
```