Big Data

Spark for streaming data

Instructor: Trong-Hop Do

September 05th 2021

S3Lab
Smart Software System Laboratory

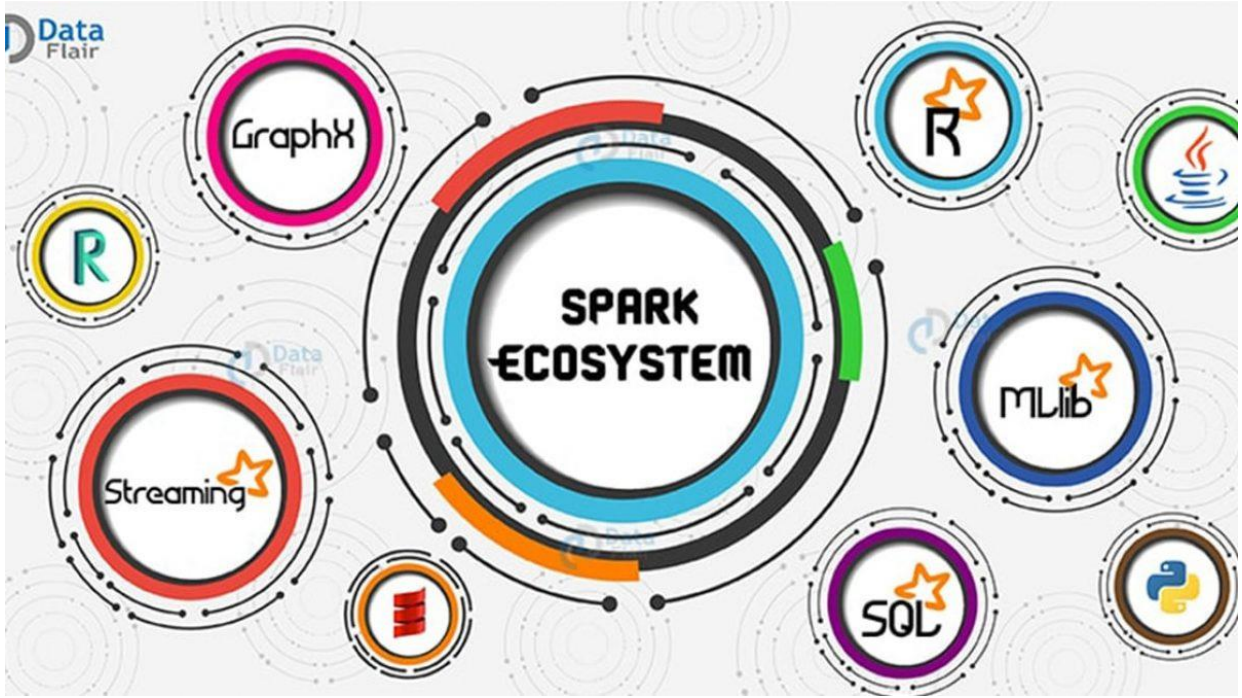
“Big data is at the foundation of all the megatrends that are happening today, from social to mobile to cloud to gaming.”

– Chris Lynch, Vertica Systems

Spark Structure Streaming

Streaming in Spark

Spark Streaming vs Structure Streaming



Streaming in Spark

Spark Streaming vs Structure Streaming

Spark

```
from pyspark.sql import streaming
```

SQL Structured Streaming

versus

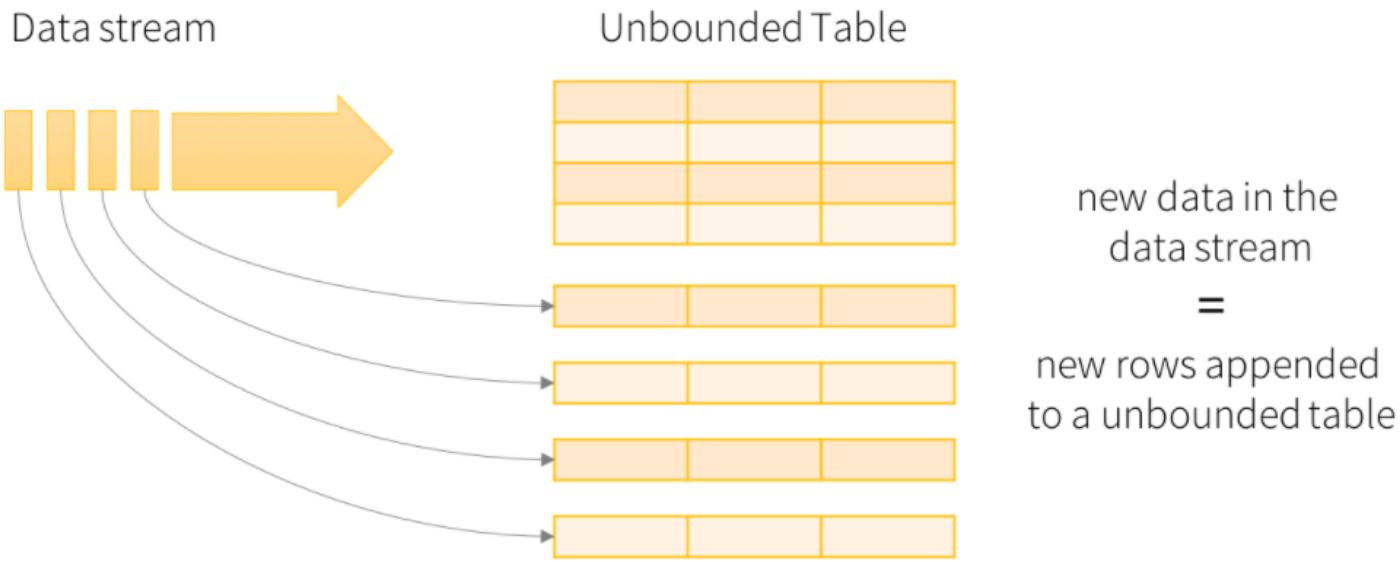
```
from pyspark import streaming
```

Streaming Component

And remember,

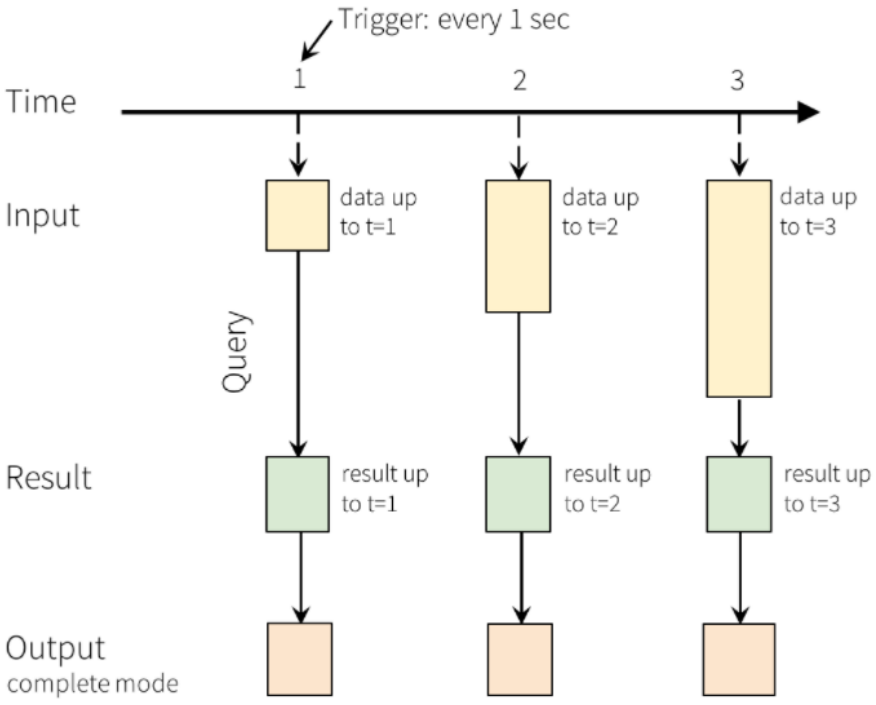
Spark SQL Structured Streaming ≠ Spark Streaming (module)

Streaming in Spark



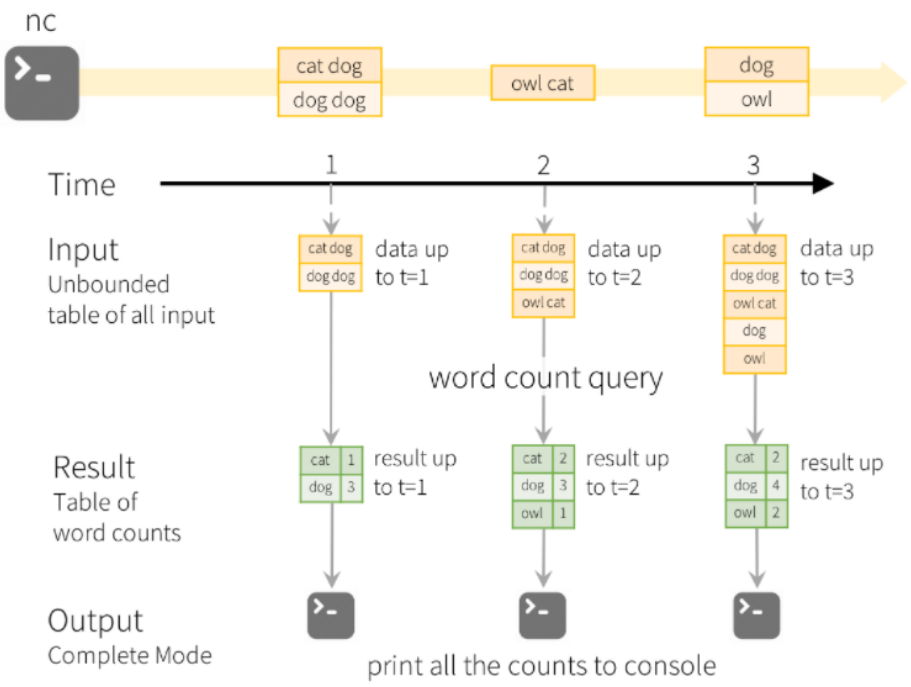
Data stream as an unbounded table

Streaming in Spark



Programming Model for Structured Streaming

Streaming in Spark



Model of the Quick Example

Streaming in Spark

Input Source

- File source
- Kafka source
- Socket source (for testing)
- Rate source (for testing)

Streaming in Spark

Output Sinks

- File sink
- Kafka sink
- Memory sink
- Console sink (for debugging)
- Foreach sink

Streaming in Spark

https://spark.apache.org/docs/latest/structured-streaming-programming-guide.html

Structure streaming

Quick example

```
import findspark
findspark.init()
import pyspark
from IPython.display import display, clear_output
from pyspark.sql import SparkSession
from pyspark.sql import functions as f
import pandas as pd
```

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import explode
from pyspark.sql.functions import split

spark = SparkSession \
    .builder \
    .appName("StructuredNetworkWordCount") \
    .getOrCreate()
```

Structure streaming

Quick example

```
# Create DataFrame representing the stream of input lines from connection to localhost:9999
lines = spark \
    .readStream \
    .format("socket") \
    .option("host", "localhost") \
    .option("port", 9999) \
    .load()

# Split the lines into words
words = lines.select(
   explode(
       split(lines.value, " ")
   ).alias("word")
)

# Generate running word count
wordCounts = words.groupBy("word").count()
```

Structure streaming

Quick example – Console sink

```
# Start running the query that prints the running counts to the console
query = wordCounts \
    .writeStream \
    .outputMode("complete") \
    .format("console") \
    .start()

query.awaitTermination()
```

Run netcat and Spark application

Quick example – Console sink

```
D:\Spark\nc112>nc -l -p 9999
cacat
dog
cat dog fish
dog fish rat
```

```
D:\Spark\spark-3.0.1-bin-hadoop2.7>spark-submit code/quickexample.py


Batch: 3
-------------------------------------------
+-----+-----+
| word|count|
+-----+-----+
|  rat|    1|
|  dog|    4|
|  cat|    1|
|cacat|    1|
| fish|    2|
+-----+-----+
```

Structure streaming

Quick example – Memory sink

```
# Start running the query that write the running counts to memory
query = wordCounts \
    .writeStream \
    .queryName("wordCounts") \
    .outputMode("complete") \
    .format("memory") \
    .start()
```

Structure streaming

Quick example – Memory sink

Command Prompt - nc -l -p 9999

```
D:\Spark\nc111nt>nc -l -p 9999
dog cat fish
dog cat
```

```
display(spark.sql(f"SELECT * from {query.name}").show())
```

```
+----+-----+
|word|count|
+----+-----+
| dog|    1|
| cat|    1|
|fish|    1|
+----+-----+
```

Structure streaming

Quick example – Memory sink

Command Prompt - nc -l -p 9999

```
D:\Spark\nc111nt>nc -l -p 9999
dog cat fish
dog cat
dog cat cat
cat cat fish
chicken
```

[7]:
```
display(spark.sql(f"SELECT * from {query.name}").show())
```

```
+----+-----+
|word|count|
+----+-----+
| dog|    1|
| cat|    1|
|fish|    1|
+----+-----+
```

None

[*]:
```
# show live results for 2 minutes, refreshed every 1 second
from time import sleep
for x in range(0, 120):
    # spark.sql can be used to request how the query is performing
    display(spark.sql(f"SELECT * from {query.name}").toPandas())
    sleep(1)
    clear_output(wait=True)
else:
    print("Live view ended...")
```

	word	count
0	dog	3
1	cat	6
2	chicken	1
3	fish	2

Structure streaming

Streaming from files

```
import pyspark
from IPython.display import display, clear_output
from pyspark.sql import SparkSession, DataFrame
from pyspark.sql import functions as f
import pandas as pd
from pyspark.ml import PipelineModel
from pyspark.sql.functions import udf
from pyspark.sql.streaming import DataStreamReader
import html

# SETTINGS
IN_PATH = "/kaggle/input/twitter-data-for-spark-streaming/"

timestampformat = "EEE MMM dd HH:mm:ss zzzz yyyy"
spark.sql("set spark.sql.legacy.timeParserPolicy=LEGACY")
spark = SparkSession.builder.appName("StructuredStreamingExample").getOrCreate()
spark.conf.set("spark.sql.legacy.timeParserPolicy","LEGACY")
schema = spark.read.json(IN_PATH).limit(10).schema


spark_reader = spark.readStream.schema(schema)
```

Structure streaming

Streaming from files

```
streaming_data_raw = (
    spark_reader.json(IN_PATH)
    .select(
        "id",
        # extract proper timestamp from created_at column
        f.to_timestamp(f.col("created_at"), timestampformat).alias("timestamp"),
        # extract user information
        f.col("user.screen_name").alias("user"),
        "text",
    )
    .coalesce(1)
)
streaming_data_clean = clean_data(streaming_data_raw)

stream_writer = (streaming_data_clean.writeStream.queryName("data").trigger(once=True).outputMode("append").format("memory"))

query = stream_writer.start()
```

Structure streaming

Streaming from files

```
display(spark.sql(f"SELECT * from {query.name}").show())
```

```
+-------------------+-------------------+---------------+--------------------+--------------------+
|                 id|          timestamp|           user|                text|       original_text|
+-------------------+-------------------+---------------+--------------------+--------------------+
|1250972456673857538|2020-04-17 02:20:50|       dibeckss|RT me after check...|RT @itsxdianaa: m...|
|1250972454035611649|2020-04-17 02:20:49| meanmediumode2|RT I love intra p...|RT @theroguecreat...|
|1250972456736526336|2020-04-17 02:20:50|       mathoeee|RT won t say i m ...|RT @ArianaGrande:...|
|1250972455625125888|2020-04-17 02:20:50|   Atalarania00|RT won t say i m ...|RT @ArianaGrande:...|
|1250972455834812416|2020-04-17 02:20:50|       kaluvbot|RT y all really o...|RT @solsticemark:...|
|1250972455457312768|2020-04-17 02:20:49|        _larrri|RT HappY Birthday...|RT @btsinluv777: ...|
|1250972453670547457|2020-04-17 02:20:49| twiliight_omen|Sorry but I won't...|Sorry, but I won'...|
|1250972453440028674|2020-04-17 02:20:49|HooverAthletics|RT Thank you for ...|RT @mj_thomas10: ...|
|1250972456061480960|2020-04-17 02:20:50| WaterBottle199|RT God Bless Amer...|RT @jakecoco: God...|
|1250972453582573572|2020-04-17 02:20:49|        tj02008|RT MS ARIANA GRAN...|RT @francisdomini...|
|1250972453695676417|2020-04-17 02:20:49|     nidzjordan|RT Today together...|RT @ADNFOREVER167...|
|1250972456493309958|2020-04-17 02:20:50|  emmaaa_blythe|RT Love my Instan...|RT @atdanwhite: L...|
|1250972456292155392|2020-04-17 02:20:50| cloutchaserprt|RT Imma go take a...|RT @RandomPocket1...|
|1250972454777786368|2020-04-17 02:20:49|      cakesiroe| RT True love's kiss|RT @Pillow_boi: T...|
|1250972454823948288|2020-04-17 02:20:49|       kthnsbno|RT Full video cov...|RT @ArianaToday: ...|
|1250972454261977089|2020-04-17 02:20:49|     sunshciner|RT You're smile m...|RT @_jenible: "Yo...|
|1250972455759462402|2020-04-17 02:20:50|sincerelyyoolie|I just love when ...|I just love when ...|
|1250972453398097920|2020-04-17 02:20:49|   yjunlionness|RT Well does that...|RT @crackheadtxt_...|
|1250972456858329088|2020-04-17 02:20:50| KballeroandanT|RT Dear we love y...|RT @That_Radish_t...|
|1250972455008493568|2020-04-17 02:20:49|        KhitLm1|RT No amount can ...|RT @MingErs_Intl:...|
+-------------------+-------------------+---------------+--------------------+--------------------+
```

Structure streaming

Streaming from files

```
distinct_user_count = streaming_data_clean.select(f.approx_count_distinct("user"), f.current_timestamp())

stream_writer = (distinct_user_count.writeStream.queryName("data").trigger(once=True).outputMode("complete").format("memory"))

query = stream_writer.start()
```

+ Code + Markdown

```
display(spark.sql(f"SELECT * from {query.name}").show())
```

```
+--------------------------+--------------------+
|approx_count_distinct(user)| current_timestamp()|
+--------------------------+--------------------+
|                        49|2021-06-20 02:47:...|
+--------------------------+--------------------+
```

Structure streaming

Streaming from files

```
sentiment_model = PipelineModel.load("/kaggle/input/pyspark-nlp/MODEL")
raw_sentiment = sentiment_model.transform(streaming_data_clean)

# Select downstream columns
sentiment = raw_sentiment.select(
    "id", "timestamp", "user", "text", f.col("prediction").alias("user_sentiment")
)
```

```
stream_writer = (sentiment.writeStream.queryName("data").trigger(once=True).outputMode("append").format("memory"))

query = stream_writer.start()
```

Structure streaming

Streaming from files

```
display(spark.sql(f"SELECT * from {query.name}").show())
```

id	timestamp	user	text	user_sentiment
1250972456673857538	2020-04-17 02:20:50	dibeckss	RT me after check...	4.0
1250972454035611649	2020-04-17 02:20:49	meanmediumode2	RT I love intra p...	4.0
1250972456736526336	2020-04-17 02:20:50	mathoeee	RT won t say i m ...	4.0
1250972455625125888	2020-04-17 02:20:50	Atalarania00	RT won t say i m ...	4.0
1250972455834812416	2020-04-17 02:20:50	kaluvbot	RT y all really o...	4.0
1250972455457312768	2020-04-17 02:20:49	_larrri	RT HappY Birthday...	4.0
1250972453670547457	2020-04-17 02:20:49	twiliight_omen	Sorry but I won't...	0.0
1250972453440028674	2020-04-17 02:20:49	HooverAthletics	RT Thank you for ...	4.0
1250972456061480960	2020-04-17 02:20:50	WaterBottle199	RT God Bless Amer...	4.0
1250972453582573572	2020-04-17 02:20:49	tj02008	RT MS ARIANA GRAN...	4.0
1250972453695676417	2020-04-17 02:20:49	nidzjordan	RT Today together...	4.0
1250972456493309958	2020-04-17 02:20:50	emmaaa_blythe	RT Love my Instan...	4.0
1250972456292155392	2020-04-17 02:20:50	cloutchaserprt	RT Imma go take a...	0.0
1250972454777786368	2020-04-17 02:20:49	cakesiroe	RT True love's kiss	4.0
1250972454823948288	2020-04-17 02:20:49	kthnsbno	RT Full video cov...	4.0
1250972454261977089	2020-04-17 02:20:49	sunshciner	RT You're smile m...	4.0
1250972455759462402	2020-04-17 02:20:50	sincerelyyoolie	I just love when ...	4.0
1250972453398097920	2020-04-17 02:20:49	yjunlionness	RT Well does that...	4.0
1250972456858329088	2020-04-17 02:20:50	KballeroandanT	RT Dear we love y...	4.0
1250972455008493568	2020-04-17 02:20:49	KhitLm1	RT No amount can ...	4.0

only showing top 20 rows

Structure streaming

Streaming from files

```
negative_sentiment_count = (
    sentiment.filter("user_sentiment == 0.0")
    .select(f.col("user_sentiment").alias("negative_sentiment"))
    .agg(f.count("negative_sentiment"))
)

positive_sentiment_count = (
    sentiment.filter("user_sentiment == 4.0")
    .select(f.col("user_sentiment").alias("positive_sentiment"))
    .agg(f.count("positive_sentiment"))
)

average_sentiment = sentiment.agg(f.avg("user_sentiment"))
```

```
data_to_stream = average_sentiment
```

Structure streaming

Streaming from files

```
if isinstance(spark_reader, DataStreamReader):
    stream_writer = (
        data_to_stream.writeStream.queryName("streaming_table")
        .trigger(processingTime="20 seconds")
        #.trigger(once=True)
        .outputMode("complete")
        .format("memory")
    )
    # Calling .start on a DataStreamWriter return an instance of StreamingQuery
    query = stream_writer.start()
```

```
display(spark.sql(f"SELECT * from {query.name}").show())
```

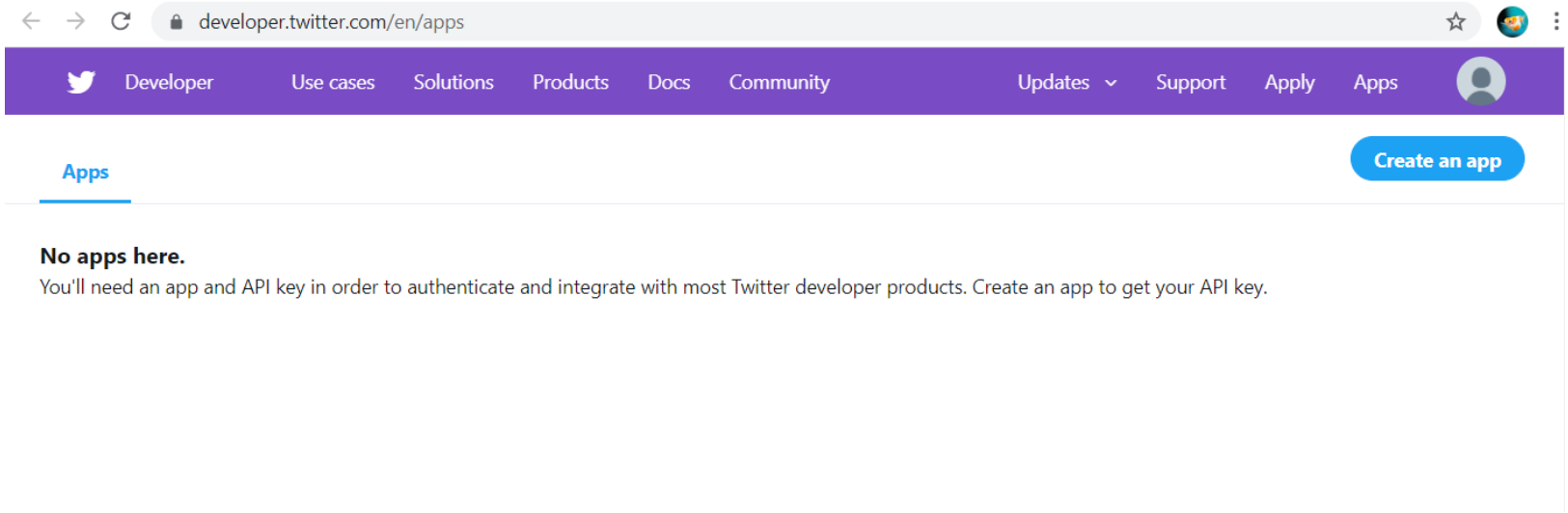
```
+-------------------+
|avg(user_sentiment)|
+-------------------+
|               3.68|
+-------------------+
```

Structure streaming

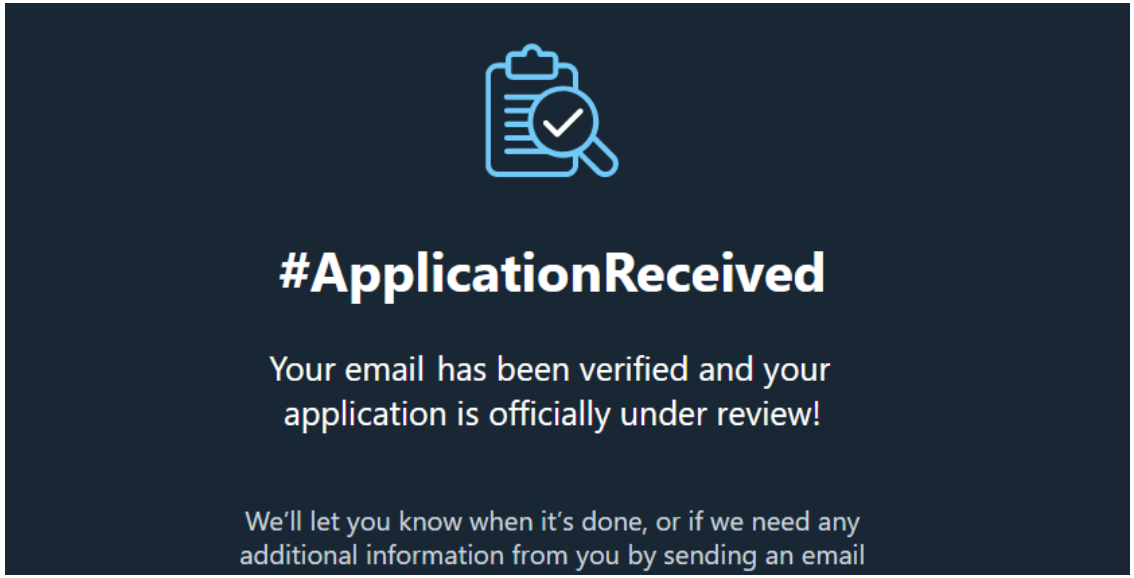
Streaming from files

```python
# Let's see what we are outputting
if streaming_data_clean.isStreaming:
    from time import sleep
    for x in range(0, 200):
        try:
            if not query.isActive:
                break
            print("Showing live view refreshed every 10 seconds")
            print(f"Seconds passed: {x*10}")
            result = spark.sql(f"SELECT * from {query.name}")
            # spark.sql can be used to request how the query is performing
            display(result.toPandas())
            sleep(10)
            clear_output(wait=True)
        except KeyboardInterrupt:
            break
    print("Live view ended...")
else:
    print("Not streaming, showing static output instead")
    result = data_to_stream
    display(result.limit(10).toPandas())
```

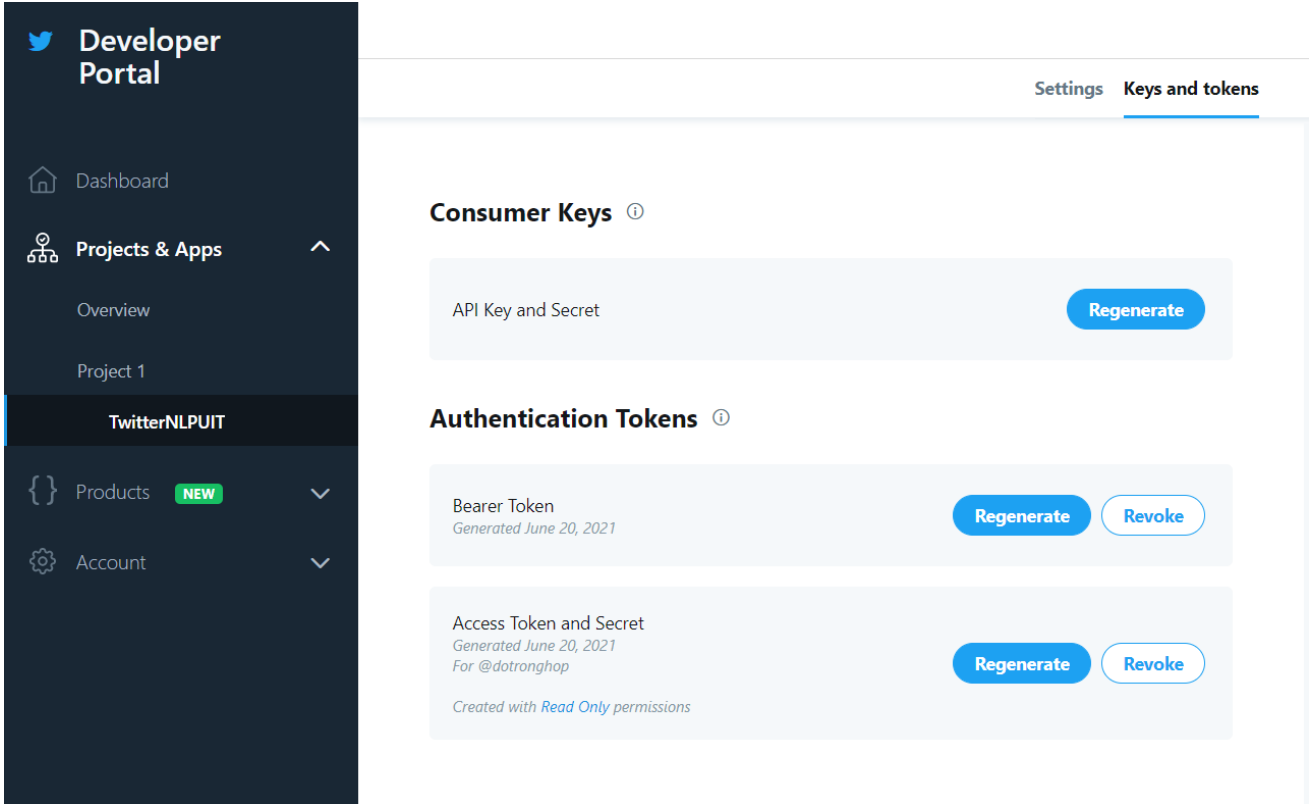
Create Twitter App



Create Twitter App



Create Twitter App



Create Twitter App

secrets.py | twitter_app.ipynb

```python
"""Replace the values below with your own Twitter API Keys, Secrets, and Tokens"""

# Twitter Consumer API keys
CONSUMER_KEY    = "0r                    dg"
CONSUMER_SECRET = "R7                                          4y"

# Twitter Access token & access token secret
ACCESS_TOKEN    = "14                                           3"
ACCESS_SECRET   = "Rj                                       1Y"


class TwitterSecrets:
    """Class that holds Twitter Secrets"""

    def __init__(self):
        self.CONSUMER_KEY    = CONSUMER_KEY
        self.CONSUMER_SECRET = CONSUMER_SECRET
        self.ACCESS_TOKEN    = ACCESS_TOKEN
        self.ACCESS_SECRET   = ACCESS_SECRET

        # Tests if keys are present
        for key, secret in self.__dict__.items():
            assert secret != "", f"Please provide a valid secret for: {key}"


twitter_secrets = TwitterSecrets()
```

Create Twitter App

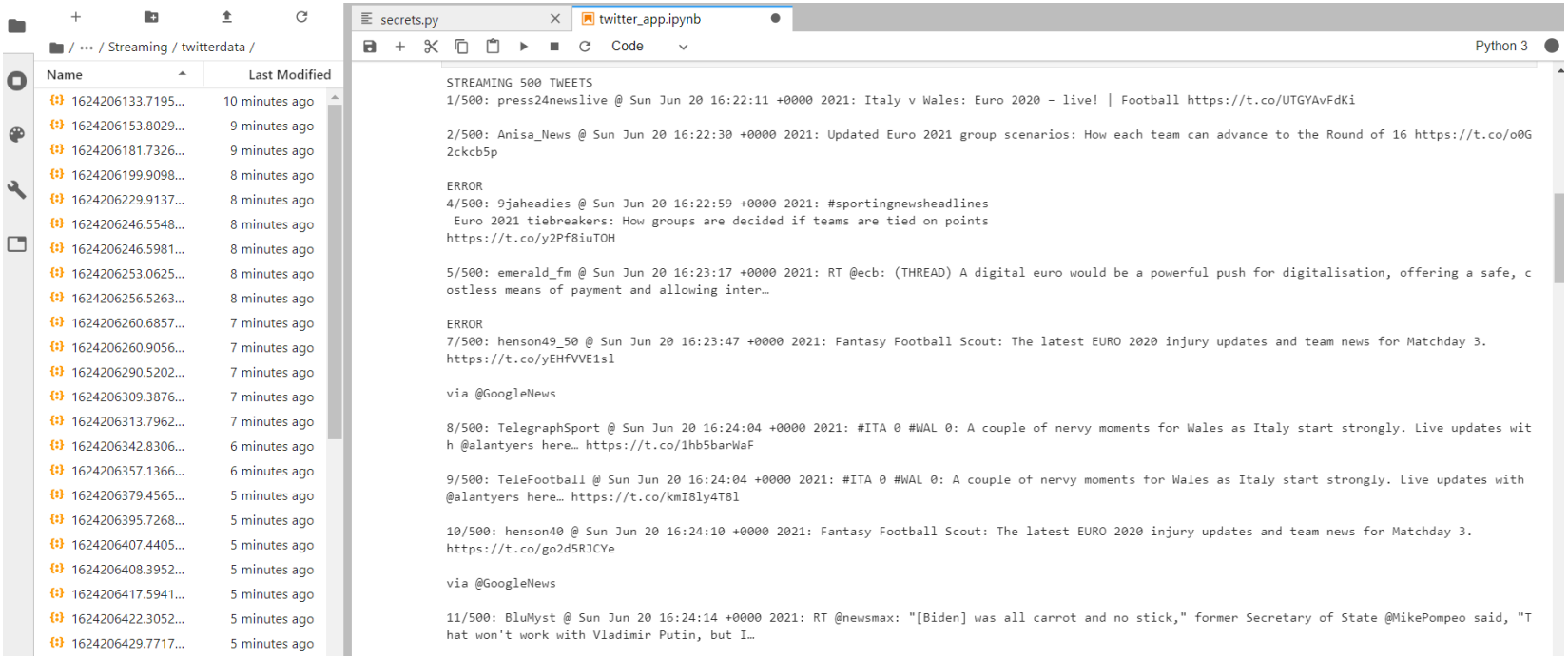
```python
from secrets import twitter_secrets as ts

OUT_PATH = "twitterdata"

QUERY = "euro 2021"

STOP_AFTER = 500
```

```python
import json
import tempfile
import requests
import pathlib
from datetime import datetime as dt
from uuid import uuid4
from requests_oauthlib import OAuth1Session
```

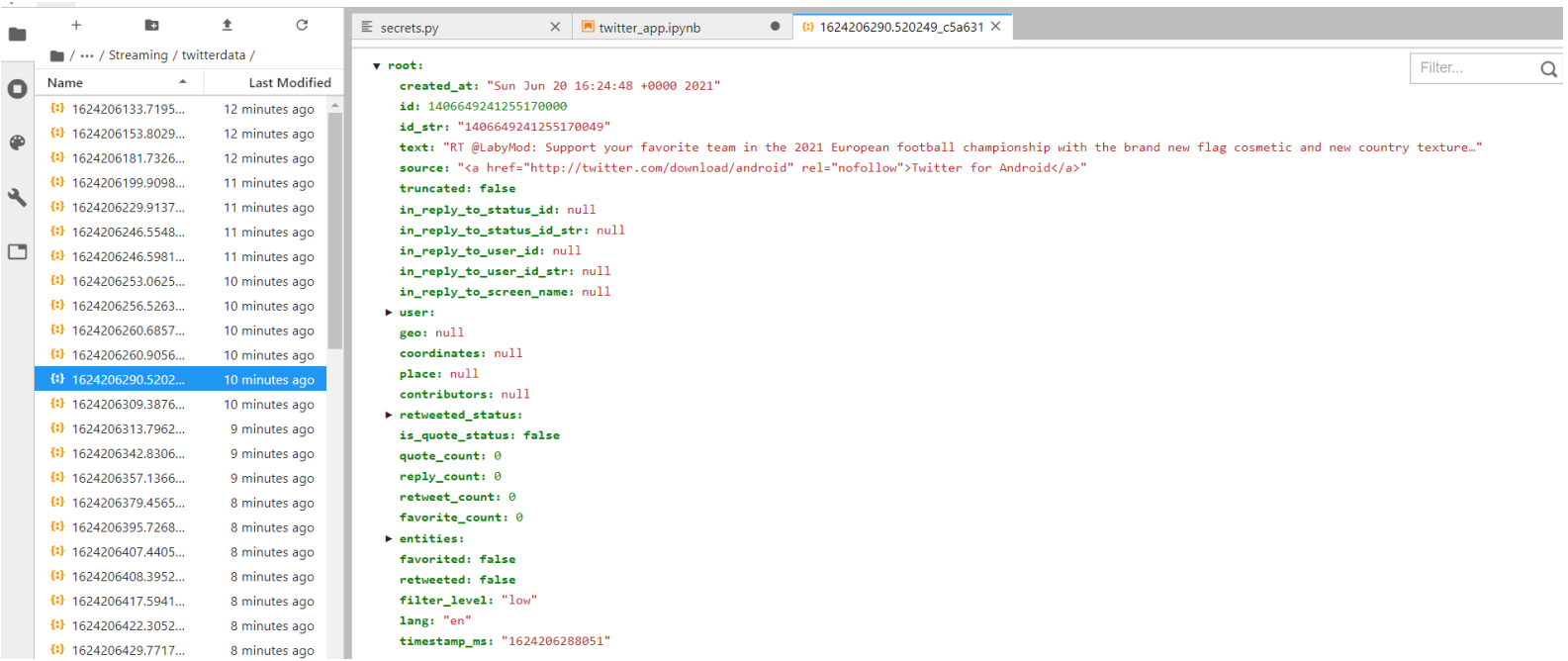
Create Twitter App

```python
pathlib.Path(OUT_PATH).mkdir(parents = True, exist_ok = True)

query_data = {
    "track": f"#{QUERY}".replace("#","").lower(), "language":"en",
}

twitter = OAuth1Session(
    client_key = ts.CONSUMER_KEY,
    client_secret = ts.CONSUMER_SECRET,
    resource_owner_key = ts.ACCESS_TOKEN,
    resource_owner_secret = ts.ACCESS_SECRET,
)

url = "https://stream.twitter.com/1.1/statuses/filter.json"
query_url = f"{url}?{'&'.join([f'{k}={v}' for k, v in query_data.items()])}"

print(f"STREAMING {STOP_AFTER} TWEETS")

with twitter.get(query_url, stream = True) as response:
    for i, raw_tweet in enumerate(response.iter_lines()):
        if i == STOP_AFTER:
            break
        try:
            tweet = json.loads(raw_tweet)
            print(f"{i+1}/{STOP_AFTER}: {tweet['user']['screen_name']} @ {tweet['created_at']}: {tweet['text']}\n")
        except (json.JSONDecodeError, KeyError) as err:
            print("ERROR")
            continue
        with pathlib.Path(OUT_PATH) / f"{dt.now().timestamp()}_{uuid4()}.json" as F:
            F.write_bytes(raw_tweet)
```

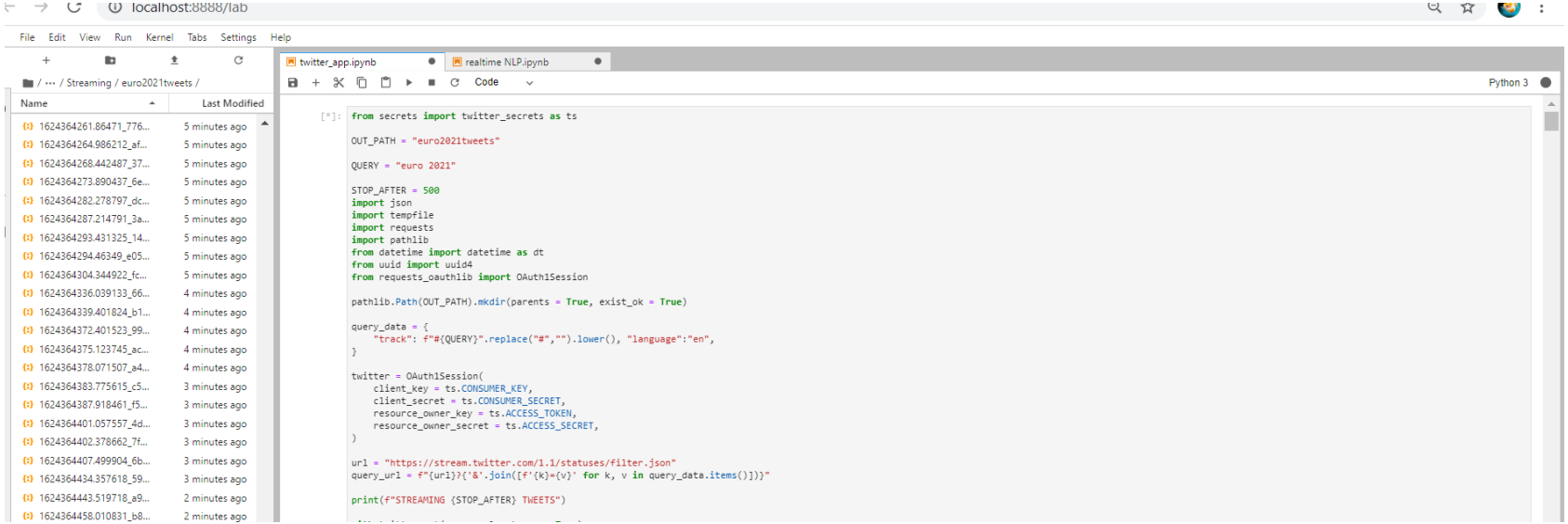
Create Twitter App



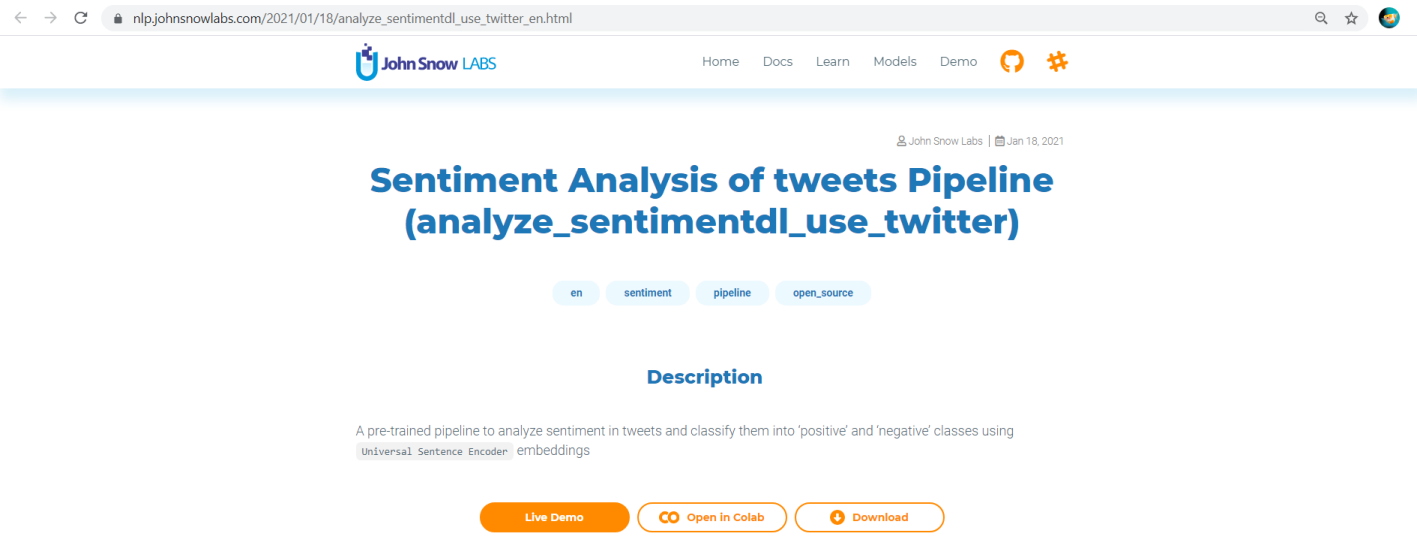
Create Twitter App



Real-time sentiment analysis



Real-time sentiment analysis



Check this

and also: https://nlp.johnsnowlabs.com/docs/en/install

Real-time sentiment analysis

```
stream_writer1 = (sentiment_result.writeStream.queryName("sentiment_result").trigger(processingTime="5 seconds").outputMode("append").format("memory"))

query1 = stream_writer1.start()

stream_writer2 = (sentiment_count_result.writeStream.queryName("data3").trigger(processingTime="5 seconds").outputMode("complete").format("memory"))

query2 = stream_writer2.start()
```

```
if streaming_data_clean.isStreaming:
    from time import sleep
    for x in range(0, 2000):
        try:
            if not query1.isActive:
                print("Query not active")
                break
            print("Showing live view refreshed every 5 seconds")
            print(f"Seconds passed: {x*5}")
            result1 = spark.sql(f"SELECT * from {query1.name}")
            result2 = spark.sql(f"SELECT * from {query2.name}")

            display(result1.toPandas())
            display(result2.toPandas())
            sleep(5)
            clear_output(wait=True)
        except KeyboardInterrupt:
            print("break")
            break
    print("Live view ended...")
else:
    print("Not streaming")
```

Real-time sentiment analysis

```
Showing live view refreshed every 5 seconds
Seconds passed: 605
```

	timestamp	user	document	sentiment
0	2021-06-22 19:18:08	barbrady1	RT So the Government wouldn t relax rules to hold the Champions league final at Wembley and sent fans to Portugal wi	negative
1	2021-06-22 18:45:44	sfsams	RT UEFA says no surprising absolutely nobody But Munich should do it anyway pay the fine Make UEFA answer all the ques	negative
2	2021-06-22 18:52:52	DuncMcKay	RT Because Public Health England a professional fucking body of experts deemed it so Grow up	negative
3	2021-06-22 18:40:53	LuluBisserier	RT Shame	neutral
4	2021-06-22 19:09:39	F1Emz	RT UEFA is a joke Acknowledging that LGBTQ people exist and deserve equal treatment is not a political statement	negative
...	...	...	...	...
285	2021-06-22 19:30:22	cfcc_connor	RT Jack Grealish to START for England tonight Praise the Lord Team news story here ENG AVFC	positive
286	2021-06-22 19:30:22	AkumiahJ	RT Jack Grealish to START for England tonight Praise the Lord Team news story here ENG AVFC	positive
287	2021-06-22 19:30:23	InnoBystander	What could possibly go wrong	negative
288	2021-06-22 19:30:24	WesthamDeku_	Saucy	positive
289	2021-06-22 19:30:25	DistinctToday	Euro Denmark s incredible moment as players find out they ve reached the last	positive

290 rows × 4 columns

	sentiment	count
0	positive	173
1	neutral	15
2	negative	93

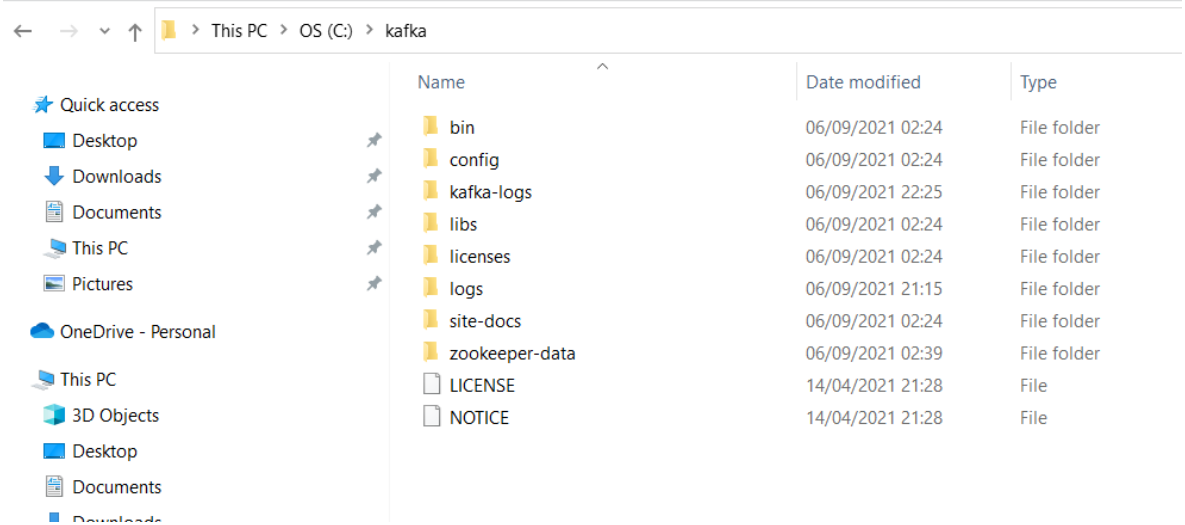
Connect Spark Structured Streaming to Kafka

- Firstly, you need to install Kafka

  https://kafka.apache.org/

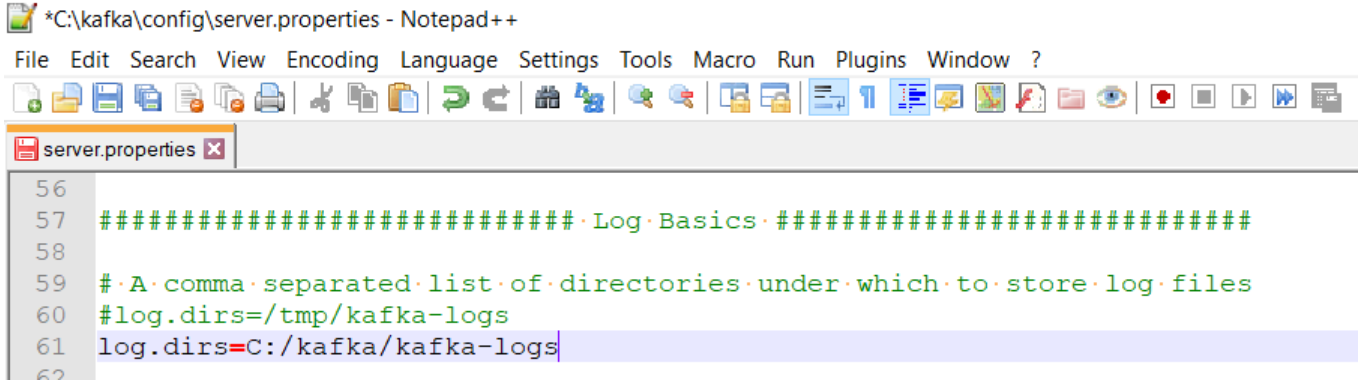
Kafka installation on Window

- Download Kafka from https://kafka.apache.org/
- Unzip the download file
- Rename the kafka to “kafka” and move it to C:\ drive



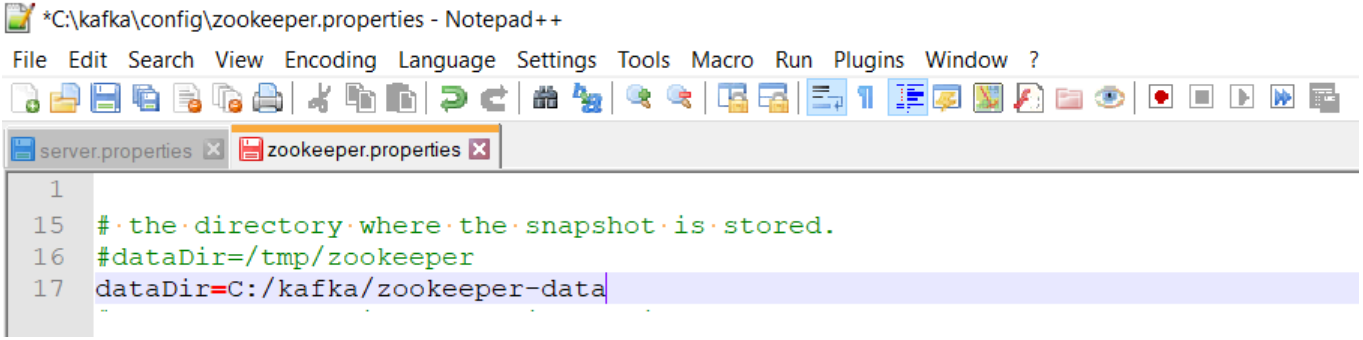
Kafka installation on Window

- Open C:\kafka\config\server.properties
- Change the path of log.dir



Kafka installation on Window

- Open C:\kafka\config\zookeeper.properties
- Change the path of dataDir



- By default Apache Kafka will run on port 9092 and Apache Zookeeper will run on port 2181.

Test Apache Kafka on Windows

- Start the Kafka cluster
  - Run the following command to start ZooKeeper:

    cd C:\kafka\

    .\bin\windows\zookeeper-server-start.bat .\config\zookeeper.properties

Start Kafka cluster - Start Zookeeper

```
C:\Users\PC>cd C:\kafka

C:\kafka>.\bin\windows\zookeeper-server-start.bat .\config\zookeeper.properties
[2021-09-06 21:58:44,055] INFO Reading configuration from: .\config\zookeeper.properties (org.apache.zookeeper.server.qu
orum.QuorumPeerConfig)
[2021-09-06 21:58:44,064] INFO clientPortAddress is 0.0.0.0:2181 (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
```

Test Apache Kafka on Windows

- Start the Kafka cluster
  - Run the following command to start the Kafka broker:

```
cd C:\kafka\
.\bin\windows\kafka-server-start.bat .\config\server.properties
```

Start Kafka cluster - Start Kafka broker

```
C:\Users\PC>cd C:\kafka

C:\kafka>.\bin\windows\kafka-server-start.bat .\config\server.properties
[2021-09-06 22:03:25,045] INFO Registered kafka:type=kafka.Log4jController MBean (kafka.utils.Log4jControllerRegistratio
n$)
```

Test Apache Kafka on Windows

- Produce and consume some messages
  - Run the kafka-topics command to create a Kafka topic named TestTopic

```
.\bin\windows\kafka-topics.bat --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic TestTopic
```

  - Let’s create another topic named NewTopic

```
.\bin\windows\kafka-topics.bat --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic NewTopic
```

  - Let’s show list of created topics

```
.\bin\windows\kafka-topics.bat --list --zookeeper localhost:2181
```

Test Apache Kafka on Windows

- Produce and consume some messages
  - Run the producer and consumer on separate Command Prompt:

.\bin\windows\kafka-console-producer.bat --broker-list localhost:9092 --topic TestTopic

.\bin\windows\kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic TestTopic --from-beginning

Producer - .\bin\windows\kafka-console-producer.bat ...

```
C:\kafka>.\bin\windows\kafka-console-producer.bat --broker-lis
t localhost:9092 --topic TestTopic
>first line
>this is the second line
>Hello World Kafka
>
```

Consumer - .\bin\windows\kafka-console-consumer.ba...

```
C:\kafka>.\bin\windows\kafka-console-consumer.bat --bootstrap-
server localhost:9092 --topic TestTopic --from-beginning
first line
this is the second line
Hello World Kafka
```

Connect Kafka and Spark Structure Streaming

- Step1: Start Kafka cluster using Terminal
- Step 2: Run KafkaProducer in Jupyter Notebook

```
from kafka import KafkaProducer
from json import dumps
from time import sleep

topic_name = 'RandomNumber'
kafka_server = 'localhost:9092'

producer = KafkaProducer(bootstrap_servers=kafka_server,value_serializer = lambda
x:dumps(x).encode('utf-8'))

for e in range(1000):
    data = {'number' : e}
    producer.send(topic_name, value=data)
    print(str(data) + " sent")
    sleep(5)

producer.flush()
```

- Open another Jupyter Notebook

[1]:
```python
import findspark
findspark.init()
import pyspark
from pyspark.sql import SparkSession

scala_version = '2.12'  # your scala version
spark_version = '3.0.1' # your spark version
packages = [
    f'org.apache.spark:spark-sql-kafka-0-10_{scala_version}:{spark_version}',
    'org.apache.kafka:kafka-clients:2.8.0' #your kafka version
]
spark = SparkSession.builder.master("local").appName("kafka-example").config("spark.jars.packages", ",".join(packages)).getOrCreate()
spark
```

[1]: **SparkSession - in-memory**

SparkContext

Spark UI

Version	`v3.0.1`
Master	`local`
AppName	`kafka-example`

- You will reading data from Kafka in two ways:
  - Batch query
  - Streaming query
- See more at https://spark.apache.org/docs/latest/structured-streaming-kafka-integration.html

Creating a Kafka Source for Batch Queries

- Create dataframe from Kafka data

```
topic_name = 'RandomNumber'
kafka_server = 'localhost:9092'
kafkaDf = spark.read.format("kafka").option("kafka.bootstrap.servers", kafka_server).option("subscribe", topic_name).option("startingOffsets",
"earliest").load()
```

- Show data (converting dataframe to pandas for cleaner view of data)

```
kafkaDf.toPandas()
```

	key	value	topic	partition	offset	timestamp	timestampType
0	None	[123, 34, 110, 117, 109, 98, 101, 114, 34, 58,...	RandomNumber	0	0	2022-10-05 15:18:37.301	0
1	None	[123, 34, 110, 117, 109, 98, 101, 114, 34, 58,...	RandomNumber	0	1	2022-10-05 15:18:42.314	0
2	None	[123, 34, 110, 117, 109, 98, 101, 114, 34, 58,...	RandomNumber	0	2	2022-10-05 15:18:47.327	0
3	None	[123, 34, 110, 117, 109, 98, 101, 114, 34, 58,...	RandomNumber	0	3	2022-10-05 15:18:52.341	0
4	None	[123, 34, 110, 117, 109, 98, 101, 114, 34, 58,...	RandomNumber	0	4	2022-10-05 15:18:57.352	0
5	None	[123, 34, 110, 117, 109, 98, 101, 114, 34, 58,...	RandomNumber	0	5	2022-10-05 15:19:02.363	0
6	None	[123, 34, 110, 117, 109, 98, 101, 114, 34, 58,...	RandomNumber	0	6	2022-10-05 15:19:07.376	0
7	None	[123, 34, 110, 117, 109, 98, 101, 114, 34, 58,...	RandomNumber	0	7	2022-10-05 15:19:12.395	0
8	None	[123, 34, 110, 117, 109, 98, 101, 114, 34, 58,...	RandomNumber	0	8	2022-10-05 15:19:17.411	0
9	None	[123, 34, 110, 117, 109, 98, 101, 114, 34, 58,...	RandomNumber	0	9	2022-10-05 15:19:22.417	0

- Show streaming data using for loop

```
batchDF = kafkaDf.select(col('topic'),col('offset'),col('value').cast('string').substr(12,1).alias('rand_number'))
from time import sleep
from IPython.display import display, clear_output
for x in range(0, 2000):
    try:
        print("Showing live view refreshed every 5 seconds")
        print(f"Seconds passed: {x*5}")
        display(batchDF.toPandas())
        sleep(5)
        clear_output(wait=True)
    except KeyboardInterrupt:
        print("break")
        break
print("Live view ended...")
```

```
Showing live view refreshed every 5 seconds
Seconds passed: 10
```

	topic	offset	rand_number
0	RandomNumber	0	2
1	RandomNumber	1	4
2	RandomNumber	2	7
3	RandomNumber	3	7
4	RandomNumber	4	3
5	RandomNumber	5	7
6	RandomNumber	6	6

- Perform some data aggregation and show live results

```python
batchCountDF = batchDF.groupBy('rand_number').count()
for x in range(0, 2000):
    try:
        print("Showing live view refreshed every 5 seconds")
        print(f"Seconds passed: {x*5}")
        display(batchCountDF.toPandas())
        sleep(5)
        clear_output(wait=True)
    except KeyboardInterrupt:
        print("break")
        break
print("Live view ended...")
```

```
Showing live view refreshed every 5 seconds
Seconds passed: 5
```

	rand_number	count
0	7	5
1	3	2
2	8	1
3	0	1
4	5	1
5	6	3
6	9	1
7	1	1
8	4	1
9	2	1

```
break
Live view ended...
```

Creating a Kafka Source for Streaming Queries

- Create Streaming dataframe from Kafka

```
streamRawDf = spark.readStream.format("kafka").option("kafka.bootstrap.servers",
kafka_server).option("subscribe", topic_name).load()
streamDF =
streamRawDf.select(col('topic'),col('offset'),col('value').cast('string').substr(12,1).alias('rand_number'))
checkEvenDF = streamDF.withColumn('Is_Even',col('rand_number').cast('int') % 2 == 0 )
```

- Write stream

```
from random import randint
randNum=str(randint(0,10000))
q1name = "queryNumber"+randNum
q2name = "queryCheckEven"+randNum

stream_writer1 = (streamDF.writeStream.queryName(q1name).trigger(processingTime="5
seconds").outputMode("append").format("memory"))
stream_writer2 = (checkEvenDF.writeStream.queryName(q2name).trigger(processingTime="5
seconds").outputMode("append").format("memory"))

query1 = stream_writer1.start()
query2 = stream_writer2.start()
```

- View streaming result

```python
for x in range(0, 2000):
    try:
        print("Showing live view refreshed every 5 seconds")
        print(f"Seconds passed: {x*5}")
        result1 = spark.sql(f"SELECT * from {query1.name}")
        result2 = spark.sql(f"SELECT * from {query2.name}")
        display(result1.toPandas())
        display(result2.toPandas())
        sleep(5)
        clear_output(wait=True)
    except KeyboardInterrupt:
        print("break")
        break
print("Live view ended...")
```

```
Showing live view refreshed every 5 seconds
Seconds passed: 20
```

	topic	offset	rand_number
0	RandomNumber	21	0
1	RandomNumber	22	8
2	RandomNumber	23	2
3	RandomNumber	24	1

	topic	offset	rand_number	Is_Even
0	RandomNumber	21	0	True
1	RandomNumber	22	8	True
2	RandomNumber	23	2	True
3	RandomNumber	24	1	False

```
break
Live view ended...
```

Q & A

Cảm ơn đã theo dõi

Chúng tôi hy vọng cùng nhau đi đến thành công.