

Load data from Kafka to Hadoop

Steps to run the python file to load data from Kafka

- Run the below command to download the Spark-SQL-Kafka jar file. This jar will be used to run the Spark Streaming-Kafka codes.

wget https://ds-spark-sql-kafka-jar.s3.amazonaws.com/spark-sql-kafka-0-10_2.11-2.3.0.jar

```
[hadoop@ip-172-31-37-94 ~]$ wget https://ds-spark-sql-kafka-jar.s3.amazonaws.com/spark-sql-kafka-0-10_2.11-2.3.0.jar
--2022-03-15 17:44:25-- https://ds-spark-sql-kafka-jar.s3.amazonaws.com/spark-sql-kafka-0-10_2.11-2.3.0.jar
Resolving ds-spark-sql-kafka-jar.s3.amazonaws.com (ds-spark-sql-kafka-jar.s3.amazonaws.com)... 52.217.230.177
Connecting to ds-spark-sql-kafka-jar.s3.amazonaws.com (ds-spark-sql-kafka-jar.s3.amazonaws.com) [52.217.230.177]:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 406313 (397K) [binary/octet-stream]
Saving to: 'spark-sql-kafka-0-10_2.11-2.3.0.jar'

100%[=====] 406,313 --.-K/s in 0.009s

2022-03-15 17:44:25 (43.1 MB/s) - 'spark-sql-kafka-0-10_2.11-2.3.0.jar' saved [406313/406313]

[hadoop@ip-172-31-37-94 ~]$ ls
spark-sql-kafka-0-10_2.11-2.3.0.jar
[hadoop@ip-172-31-37-94 ~]$
```

- Below is the screenshot of the python file (**spark_kafka_to_local.py**) containing the code to ingest the relevant data from Kafka and load the data into hadoop file system.

```
[hadoop@ip-172-31-37-94 ~]$ cat spark_kafka_to_local.py
# Importing the SparkSession from Pyspark's SQL module
from pyspark.sql import SparkSession

# Initializing the Spark session
# Providing a meaningful name to the application relevant to the steps performed in the application
# Set the log level to error
spark = SparkSession \
    .builder \
    .appName("ClickstreamDatatoHadoop") \
    .getOrCreate()
spark.sparkContext.setLogLevel('ERROR')

# Reading the data from Kafka topic
# startingOffsets used as earliest to load all the data from the beginnng
# Provided the given Kafka server along with port
# Provided the Kafka topic name from which the data has to be read
rawData = spark \
    .readStream \
    .format("kafka") \
    .option("multiline","true") \
    .option("startingOffsets", "earliest") \
    .option("kafka.bootstrap.servers","18.211.252.152:9092") \
    .option("subscribe","de-capstone3") \
    .load()

data = rawData.selectExpr("CAST(key AS STRING)", "CAST(value AS STRING)")

# Writing the raw data to the Hadoop location in JSON format
# Provided target directory in Hadoop where the data will be stored
# Provided target directory in Hadoop where the data checkpoint will be created
finalClickStream = data \
    .writeStream \
    .format("json") \
    .outputMode("append") \
    .option("truncate", "false") \
    .option('path', "/user/hadoop/clickStreamData") \
    .option("checkpointLocation", "/user/hadoop/clickStreamDataCP") \
    .start() \
    .awaitTermination()
[hadoop@ip-172-31-37-94 ~]$
```

- Below is the spark-submit command to run the **spark_kafka_to_local.py** file.

Command: spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.11:2.4.5
spark_kafka_to_local.py ###.####.####.### ##### de-capstone3

- Below is the screenshot of the data imported from Kafka to Hadoop.

```
[hadoop@ip-172-31-37-94 ~]$ hadoop fs -ls /user/hadoop/clickStreamData
Found 2 items
drwxr-xr-x   - hadoop hadoop          0 2022-03-15 17:50 /user/hadoop/clickStreamData/_spark_metadata
-rw-r--r--   1 hadoop hadoop    1255605 2022-03-15 17:50 /user/hadoop/clickStreamData/part-00000-0d1fa81d-8fc9-4735-bd94-7bf9a96552f8-c000.json
[hadoop@ip-172-31-37-94 ~]$ hadoop fs -cat /user/hadoop/clickStreamData/part-00000-0d1fa81d-8fc9-4735-bd94-7bf9a96552f8-c000.json
{"value":{"customer_id": "26544820", "app_version": "3.2.35", "OS version": "Android", "lat": "16.4454865", "lon": "99.902065", "page_id": "de54571", "button_id": "fcba68aa-1231-11eb-adc1-0242ac120002", "is_button_click": "No", "is_page_view": "Yes", "is_scroll_up": "No", "is_scroll_down": "Yes", "timestamp": "2020-09-14 09:59:07\\n\\n"}}
{"value":{"customer_id": "31906397", "app_version": "2.4.7", "OS version": "iOS", "lat": "-64.813749", "lon": "-133.527040", "page_id": "de54571", "button_id": "a95dd57b-775f-49db-819d-d6960483e554", "is_button_click": "No", "is_page_view": "No", "is_scroll_up": "Yes", "is_scroll_down": "Yes", "timestamp": "2020-05-16 16:30:21\\n\\n"}}
{"value":{"customer_id": "25713677", "app_version": "3.4.12", "OS version": "Android", "lat": "89.943435", "lon": "127.313415", "page_id": "b328", "button_id": "fcba68aa-1231-11eb-adc1-0242ac120002", "is_button_click": "No", "is_page_view": "No", "is_scroll_up": "Yes", "is_scroll_down": "No", "timestamp": "2020-02-09 00:52:13\\n\\n"}}
{"value":{"customer_id": "83474293", "app_version": "3.1.8", "OS version": "Android", "lat": "-69.939070", "lon": "-36.451670", "page_id": "e7bc", "button_id": "ele99492-17ae-11eb-adc1-0242ac120002", "is_button_click": "Yes", "is_page_view": "No", "is_scroll_up": "Yes", "is_scroll_down": "No", "timestamp": "2020-06-17 10:42:50\\n\\n"}}
{"value":{"customer_id": "63727807", "app_version": "2.2.9", "OS version": "iOS", "lat": "64.082108", "lon": "-81.822078", "page_id": "e7bc5fb2", "button_id": "fcba68aa-1231-11eb-adc1-0242ac120002", "is_button_click": "No", "is_page_view": "Yes", "is_scroll_up": "Yes", "is_scroll_down": "Yes", "timestamp": "2020-07-06 02:51:53\\n\\n"}}
{"value":{"customer_id": "73737907", "app_version": "4.3.19", "OS version": "Android", "lat": "-18.850508", "lon": "-116.358375", "page_id": "b3", "button_id": "ele99492-17ae-11eb-adc1-0242ac120002", "is_button_click": "No", "is_page_view": "Yes", "is_scroll_up": "No", "is_scroll_down": "Yes", "timestamp": "2020-04-26 06:18:16\\n\\n"}}
```

Now we will run the **spark_local_flatten.py** python file to clean the loaded Kafka data to a more structured format.

- Below is the screenshot of the python file (**spark_local_flatten.py**) containing the code to clean the loaded Kafka data to a more structured format.

```
[hadoop@ip-172-31-35-157 ~]$ cat spark_local_flatten.py
# Importing the SparkSession, functions and types from Pyspark's SQL module
from pyspark.sql import SparkSession
from pyspark.sql.functions import *
from pyspark.sql.types import StringType, IntegerType, FloatType, TimestampType
# Initializing the Spark session
# Providing a meaningful name to the application relevant to the steps performed in the application
# Set the log level to error
spark = SparkSession \
    .builder \
    .appName("StructuredClickstreamDataToHadoop") \
    .getOrCreate()
spark.sparkContext.setLogLevel('ERROR')
# Reading the data from JSON file into a dataframe
df = spark.read.json('/user/hadoop/clickStreamData/*.json')
# Creating a new dataframe by extracting the columns from the JSON value
df1 = df.select(get_json_object(df['value'], "$.customer_id").alias("customer_id"),
               get_json_object(df['value'], "$.app_version").alias("app_version"),
               get_json_object(df['value'], "$.OS version").alias("OS version"),
               get_json_object(df['value'], "$.lat").alias("lat"),
               get_json_object(df['value'], "$.lon").alias("lon"),
               get_json_object(df['value'], "$.page_id").alias("page_id"),
               get_json_object(df['value'], "$.button_id").alias("button_id"),
               get_json_object(df['value'], "$.is_button_click").alias("is_button_click"),
               get_json_object(df['value'], "$.is_page_view").alias("is_page_view"),
               get_json_object(df['value'], "$.is_scroll_up").alias("is_scroll_up"),
               get_json_object(df['value'], "$.is_scroll_down").alias("is_scroll_down"),
               get_json_object(df['value'], "$.timestamp\\n").alias("timestamp"))
# Extracting only the required part from the timestamp column and saving in new column "timestamp"
df2 = df1.withColumn("timestamp", df1["timestamp"][0:19])
# Creating a new dataframe by assigning the correct datatypes to the "customer_id", "lat" and "lon" columns
df3 = df2 \
    .withColumn("customer_id", df2["customer_id"].cast(IntegerType())) \
    .withColumn("lat", df2["lat"].cast(FloatType())) \
    .withColumn("lon", df2["lon"].cast(FloatType()))
# Selecting only the required columns as per the given schema
df4 = df3.select("customer_id", "app_version", "OS version", "lat", "lon", "page_id", "button_id", "is_button_click", "is_page_view", "is_scroll_up", "is_scroll_down", "timestamp")
# Printing the schema of the dataframe
print(df4.schema)
# Printing the first 10 records from the dataframe
df4.show(5)
# Printing the number of records
records = df4.count()
print("Number of records = " + str(records))
# Writing the structured clickstream data in CSV format to the hadoop location
df4.coalesce(1).write.format('csv').option('header', 'false').save('/user/hadoop/structuredclickStreamData', mode='overwrite')
[hadoop@ip-172-31-35-157 ~]$
```

- Below is the spark-submit command to run the **spark_local_flatten.py** file.

Command: spark-submit spark_local_flatten.py

- Below is the screenshot of the output after running the **spark_local_flatten.py** file.

```
22/03/16 14:38:53 INFO StateStoreCoordinatorRef: Registered StateStoreCoordinator endpoint
StructType(List(StructField(customer_id,IntegerType,true),StructField(app_version,StringType,true),StructField(OS_version,StringType,true),StructField(lat,FloatType,true),StructField(lon,FloatType,true),StructField(page_id,StringType,true),StructField(button_id,StringType,true),StructField(is_button_click,StringType,true),StructField(is_page_view,StringType,true),StructField(is_scroll_up,StringType,true),StructField(is_scroll_down,StringType,true),StructField(timestamp,StringType,true)))
-----+-----+
|customer_id|app_version|OS_version|lat|lon|page_id|button_id|is_button_click|is_page_view|is_scroll_up|is_scroll_down|timestamp|
-----+-----+
|26564820|3.2.35|Android|16.445486|99.90206|de545711-3914-4450-8c11-b17b8dabb5e1|fcba68aa-1231-11eb-adc1-0242ac120002|No|Yes|No|Yes|2020-09-14 09:59:07|
|31906387|2.4.7|iOS|-64.81375|-133.52704|de545711-3914-4450-8c11-b17b8dabb5e1|a95dd57b-779f-49db-819d-b6960483e554|No|No|Yes|Yes|2020-05-16 16:30:21|
|25713677|3.4.12|Android|89.943436|127.313416|b328829e-17ae-11eb-adc1-0242ac120002|fcba68aa-1231-11eb-adc1-0242ac120002|No|No|Yes|No|2020-02-09 00:52:13|
|83474293|3.1.8|Android|-69.93907|-36.45167|e7bc5fb2-1231-11eb-adc1-0242ac120002|e1e99492-17ae-11eb-adc1-0242ac120002|Yes|No|Yes|No|2020-06-17 10:42:50|
|63727807|2.2.9|iOS|64.08211|-81.822075|e7bc5fb2-1231-11eb-adc1-0242ac120002|fcba68aa-1231-11eb-adc1-0242ac120002|No|Yes|Yes|Yes|2020-07-06 02:51:53|
only showing top 5 rows
Number of records = 3000
[hadoop@ip-172-31-35-157 ~]$
```

Steps to load the data into Hadoop

Data is loaded into HDFS at “/user/hadoop/structuredclickStreamData” directory using the below command which is included in the **spark_local_flatten.py** file.

Command:

df4.coalesce(1).write.format('csv').option('header','false').save('/user/hadoop/structuredclickStreamData', mode='overwrite')

Screenshot of the data

- Below is the screenshot of the cleaned and structured data loaded in the Hadoop in CSV format. (**/user/hadoop/structuredclickStreamData**)

```
[hadoop@ip-172-31-35-157 ~]$ hadoop fs -ls /user/hadoop/structuredclickStreamData
Found 2 items
-rw-r--r-- 1 hadoop hadoop 0 2022-03-16 14:39 /user/hadoop/structuredclickStreamData/_SUCCESS
-rw-r--r-- 1 hadoop hadoop 449608 2022-03-16 14:39 /user/hadoop/structuredclickStreamData/part-00000-94dfbdc7-6df0-4807-959b-69dabaeb2081-c000.csv
[hadoop@ip-172-31-35-157 ~]$ hadoop fs -cat /user/hadoop/structuredclickStreamData/part-00000-94dfbdc7-6df0-4807-959b-69dabaeb2081-c000.csv
26564820,3.2.35,Android,16.445486,99.90206,de545711-3914-4450-8c11-b17b8dabb5e1,fcba68aa-1231-11eb-adc1-0242ac120002,No,Yes,No,Yes,2020-09-14 09:59:07
31906387,2.4.7,iOS,-64.81375,-133.52704,de545711-3914-4450-8c11-b17b8dabb5e1,a95dd57b-779f-49db-819d-b6960483e554,No,No,Yes,Yes,2020-05-16 16:30:21
25713677,3.4.12,Android,89.943436,127.313416,b328829e-17ae-11eb-adc1-0242ac120002,fcba68aa-1231-11eb-adc1-0242ac120002,No,No,Yes,No,2020-02-09 00:52:13
83474293,3.1.8,Android,-69.93907,-36.45167,e7bc5fb2-1231-11eb-adc1-0242ac120002,e1e99492-17ae-11eb-adc1-0242ac120002,Yes,No,Yes,No,2020-06-17 10:42:50
63727807,2.2.9,iOS,64.08211,-81.822075,e7bc5fb2-1231-11eb-adc1-0242ac120002,fcba68aa-1231-11eb-adc1-0242ac120002,No,Yes,Yes,Yes,2020-07-06 02:51:53
73737907,4.3.19,Android,-18.850508,-116.358376,b328829e-17ae-11eb-adc1-0242ac120002,e1e99492-17ae-11eb-adc1-0242ac120002,No,Yes,No,Yes,2020-04-26 06:18:16
36927433,3.2.26,iOS,-84.68572,-146.50768,de545711-3914-4450-8c11-b17b8dabb5e1,a95dd57b-779f-49db-819d-b6960483e554,Yes,Yes,No,Yes,2020-02-06 10:21:18
12691783,3.3.11,Android,54.385292,-37.411816,de545711-3914-4450-8c11-b17b8dabb5e1,e1e99492-17ae-11eb-adc1-0242ac120002,Yes,Yes,No,No,2020-08-08 04:23:56
22635021,4.4.36,iOS,-31.8055,150.65565,e7bc5fb2-1231-11eb-adc1-0242ac120002,a95dd57b-779f-49db-819d-b6960483e554,No,No,No,2020-08-02 00:33:50
23593546,1.2.16,Android,8.891848,-83.92988,de545711-3914-4450-8c11-b17b8dabb5e1,e1e99492-17ae-11eb-adc1-0242ac120002,Yes,No,Yes,No,2020-07-23 23:59:19
16929816,1.2.25,Android,-34.85764,-136.63983,e7bc5fb2-1231-11eb-adc1-0242ac120002,fcba68aa-1231-11eb-adc1-0242ac120002,No,No,No,Yes,2020-05-13 16:35:17
41929865,1.1.14,iOS,-34.361317,-137.46783,b328829e-17ae-11eb-adc1-0242ac120002,a95dd57b-779f-49db-819d-b6960483e554,No,Yes,No,No,2020-04-13 07:58:26
68940320,1.1.17,Android,81.738235,-104.20776,de545711-3914-4450-8c11-b17b8dabb5e1,fcba68aa-1231-11eb-adc1-0242ac120002,Yes,No,Yes,No,2020-10-14 00:39:05
95405928,1.2.10,Android,44.185627,78.12549,e7bc5fb2-1231-11eb-adc1-0242ac120002,e1e99492-17ae-11eb-adc1-0242ac120002,No,No,No,No,2020-07-03 19:15:34
56235860,1.4.16,Android,-61.178406,8.534126,de545711-3914-4450-8c11-b17b8dabb5e1,a95dd57b-779f-49db-819d-b6960483e554,Yes,Yes,Yes,Yes,2020-06-02 21:32:44
81758845,4.4.30,Android,78.90498,167.3156,e7bc5fb2-1231-11eb-adc1-0242ac120002,a95dd57b-779f-49db-819d-b6960483e554,Yes,Yes,No,No,2020-04-17 19:52:56
46822067,1.2.26,Android,45.8994,135.89511,de545711-3914-4450-8c11-b17b8dabb5e1,fcba68aa-1231-11eb-adc1-0242ac120002,Yes,Yes,Yes,Yes,2020-08-09 12:02:09
37623067,3.2.27,iOS,-74.11315,-170.71542,b328829e-17ae-11eb-adc1-0242ac120002,a95dd57b-779f-49db-819d-b6960483e554,No,Yes,No,No,2020-01-20 00:06:06
14571693,3.4.21,iOS,-28.832905,161.13237,e7bc5fb2-1231-11eb-adc1-0242ac120002,fcba68aa-1231-11eb-adc1-0242ac120002,No,Yes,No,No,2020-08-16 11:20:40
62605529,2.1.36,Android,44.19624,-18.354515,e7bc5fb2-1231-11eb-adc1-0242ac120002,a95dd57b-779f-49db-819d-b6960483e554,Yes,Yes,No,No,2020-06-24 01:55:32
83657253,2.1.2,iOS,38.7036,-59.08193,b328829e-17ae-11eb-adc1-0242ac120002,fcba68aa-1231-11eb-adc1-0242ac120002,No,Yes,No,Yes,2020-05-07 20:53:51
```