FINISHED

dsti-a19/ref/lab4

How to submit the Spark application:

1. SSH to the edge server
2. Get the code from HDFS

```
hdfs dfs -get /learning/code/spark .
```

3. Cd to the code folder: `cd spark/application-submitting`
4. Check out the documentation at https://github.com/adaltas/ece-spark/tree/master/application-submitting (https://github.com/adaltas/ece-spark/tree/master/application-submitting)

Took 0 sec. Last updated by gauthier at March 18 2020, 4:24:16 PM.

FINISHED

```
import argparse
import getpass

from pyspark.sql import SparkSession
from pyspark.sql.functions import explode, split, window, count, mean

# Use Python argparse library to parse arguments
parser = argparse.ArgumentParser(
    description='Process NYC Taxi datasets in streaming using sockets')
parser.add_argument('appname', type=str, help='The Spark application name')
parser.add_argument('sockethostname', type=str,
                    help='The hostname on which to listen to the socket')
parser.add_argument('outputpath', type=str,
                    help='The HDFS path where to write the CSV output')
parser.add_argument('-f', '--faresport', type=int, default=11111,
                    help='The port on which the fares dataset is streamed')
parser.add_argument('-r', '--ridesport', type=int, default=11112,
                    help='The port on which the rides dataset is streamed')
parser.add_argument('-c', '--checkpoint', type=str, help='The HDFS path '
                    'where Spark will write checkpointing infomation. '
                    'Default = /user/USERNAME/checkpoint/APP_NAME')

args = parser.parse_args()

# Create a new SparkSession
spark = SparkSession \
    .builder \
    .appName(args.appname) \
    .getOrCreate()

# Print the application Web UI url
print('Application Web UI: %s' % spark.sparkContext.uiWebUrl)

# Define the stream source (socket)
fares_raw = spark \
    .readStream \
    .format("socket") \
    .option("host", args.sockethostname) \
    .option("port", args.faresport) \
    .load()

# Parse the socket message "manually"
fares = fares_raw \
    .select(
        split(fares_raw.value, ',')[0].alias('ride_id').cast('int'),
        split(fares_raw.value, ',')[1].alias('taxi_id').cast('int'),
        split(fares_raw.value, ',')[2].alias('driver_id').cast('int'),
        split(fares_raw.value, ',')[3].alias('start_time').cast('timestamp'),
        split(fares_raw.value, ',')[4].alias('payment_type'),
        split(fares_raw.value, ',')[5].alias('tip').cast('float'),
        split(fares_raw.value, ',')[6].alias('tolls').cast('float'),
        split(fares_raw.value, ',')[7].alias('total_fare').cast('float')
    ) \
    .withWatermark('start_time', '1 minutes') \

# Define the aggregation to perform on the stream
fares_count = fares \
```

```
    .withWatermark('start_time', '1 minutes') \
    .groupBy(window(fares.start_time, '1 minutes', '1 minutes')) \
    .agg(
        count('ride_id').alias('ride_count'),
        mean('total_fare').alias('mean_total_fare')
    )

# Define the stream sick (parquet files written to HDFS)
# The trigger control the occurance of parquet file generation
fares_count_query = fares_count \
    .writeStream \
    .outputMode('append') \
    .format('parquet') \
    .trigger(processingTime='10 seconds') \
    .option('path', args.outputpath) \
    .option('checkpointLocation', args.checkpoint if args.checkpoint
            else '/user/{}/checkpoint/{}'.format(
                getpass.getuser(), args.appname)) \
    .start()

# Wait for the end of the query before ending the application
fares_count_query.awaitTermination()
```


Took 0 sec. Last updated by gauthier at March 18 2020, 3:26:58 PM.

```
%pyspark
nyxtaxi_app_result = spark.read.parquet('/user/gauthier/spark-streaming-demo-dsti')
z.show(nyxtaxi_app_result)
```

FINISHED

settings

window	ride_count
[2020-03-18 14:34:00.0,2020-03-18 14:35:00.0]	92
[2020-03-18 14:35:00.0,2020-03-18 14:36:00.0]	115
[2020-03-18 14:36:00.0,2020-03-18 14:37:00.0]	143
[2020-03-18 14:37:00.0,2020-03-18 14:38:00.0]	186
[2020-03-18 14:38:00.0,2020-03-18 14:39:00.0]	199
[2020-03-18 14:39:00.0,2020-03-18 14:40:00.0]	259
[2020-03-18 14:40:00.0,2020-03-18 14:41:00.0]	337

Took 1 sec. Last updated by gauthier at March 18 2020, 3:42:29 PM.

```
%jdbc
```

READY