project_streaming

October 4, 2024

[1]:
```
# Import SparkSession
from pyspark.sql import SparkSession
# Create a Spark Session
spark = SparkSession.builder.master("local[*]").getOrCreate()
# Check Spark Session Information
```

```
24/10/04 10:04:52 WARN Utils: Your hostname, MacBook-Pro-dAudric.local resolves
to a loopback address: 127.0.0.1; using 192.168.209.155 instead (on interface
en0)
24/10/04 10:04:52 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another
address
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use
setLogLevel(newLevel).
24/10/04 10:04:52 WARN NativeCodeLoader: Unable to load native-hadoop library
for your platform… using builtin-java classes where applicable
```

0.1 Streaming with socket

[3]:
```
from pyspark.sql.functions import explode, split, current_timestamp, window

port = 9999

# df spark représentant le flux d'entrée à partir de la connexion socket
stream = spark.readStream \
    .format("socket") \
    .option("host", "localhost") \
    .option("port", port) \
    .load()\
    .withColumn("timestamp", current_timestamp()) \

#Utilisation de fenêtres temporelles
w = stream.withColumn("window", window("timestamp", "1 minute"))

# Définir la fonction qui traite chaque batch de données
def split_words(df, batch_id):
    # Séparer les lignes en mots et sauvegarder le résultat dans une table Spark
    words = df.withColumn("word", explode(split(df["value"], " ")))
```

```
    # Sauvegarder le DataFrame dans une table Spark
    words.write \
        .mode("append") \
        .saveAsTable("array_words")

# Afficher les résultats en continu
query = w.writeStream \
            .foreachBatch(split_words) \
            .outputMode("append") \
            .start()

# Attendre que la requête se termine
query.awaitTermination()
```

```
24/10/04 10:05:09 WARN TextSocketSourceProvider: The socket source should not be
used for production applications! It does not support recovery.
24/10/04 10:05:09 WARN ResolveWriteToStream: Temporary checkpoint location
created which is deleted normally when the query didn't fail: /private/var/folde
rs/q7/05h3nfts6n3_246h_khmw2z00000gn/T/temporary-49556978-fa02-4e02-bf52-
f30dab652aca. If it's required to delete it under any circumstances, please set
spark.sql.streaming.forceDeleteTempCheckpointLocation to true. Important to know
deleting temp checkpoint folder is best effort.
24/10/04 10:05:09 WARN ResolveWriteToStream: spark.sql.adaptive.enabled is not
supported in streaming DataFrames/Datasets and will be disabled.
24/10/04 10:10:29 WARN TextSocketMicroBatchStream: Stream closed by
localhost:9999
ERROR:root:KeyboardInterrupt while sending command.
Traceback (most recent call last):
  File "/Users/aaudric/miniconda3/envs/pyspark_env/lib/python3.12/site-
packages/py4j/java_gateway.py", line 1038, in send_command
    response = connection.send_command(command)
               ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

  File "/Users/aaudric/miniconda3/envs/pyspark_env/lib/python3.12/site-
packages/py4j/clientserver.py", line 511, in send_command
    answer = smart_decode(self.stream.readline()[:-1])
                          ^^^^^^^^^^^^^^^^^^^^^^^

  File "/Users/aaudric/miniconda3/envs/pyspark_env/lib/python3.12/socket.py",
line 720, in readinto
    return self._sock.recv_into(b)
           ^^^^^^^^^^^^^^^^^^^^^^^

KeyboardInterrupt
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
Cell In[3], line 33
     27 query = w.writeStream \
```

```
     28             .foreachBatch(split_words) \
     29             .outputMode("append") \
     30             .start()
     32 # Attendre que la requête se termine
---> 33 query.awaitTermination()

File ~/miniconda3/envs/pyspark_env/lib/python3.12/site-packages/pyspark/sql/
 ↪streaming/query.py:221, in StreamingQuery.awaitTermination(self, timeout)
    219     return self._jsq.awaitTermination(int(timeout * 1000))
    220 else:
--> 221     return self._jsq.awaitTermination()

File ~/miniconda3/envs/pyspark_env/lib/python3.12/site-packages/py4j/
 ↪java_gateway.py:1321, in JavaMember.__call__(self, *args)
   1314 args_command, temp_args = self._build_args(*args)
   1316 command = proto.CALL_COMMAND_NAME +\
   1317     self.command_header +\
   1318     args_command +\
   1319     proto.END_COMMAND_PART
-> 1321 answer = self.gateway_client.send_command(command)
   1322 return_value = get_return_value(
   1323     answer, self.gateway_client, self.target_id, self.name)
   1325 for temp_arg in temp_args:

File ~/miniconda3/envs/pyspark_env/lib/python3.12/site-packages/py4j/
 ↪java_gateway.py:1038, in GatewayClient.send_command(self, command, retry,␣
 ↪binary)
   1036 connection = self._get_connection()
   1037 try:
-> 1038     response = connection.send_command(command)
   1039     if binary:
   1040         return response, self._create_connection_guard(connection)

File ~/miniconda3/envs/pyspark_env/lib/python3.12/site-packages/py4j/
 ↪clientserver.py:511, in ClientServerConnection.send_command(self, command)
    509 try:
    510     while True:
--> 511         answer = smart_decode(self.stream.readline()[:-1])
    512         logger.debug("Answer received: {0}".format(answer))
    513         # Happens when a the other end is dead. There might be an empty
    514         # answer before the socket raises an error.

File ~/miniconda3/envs/pyspark_env/lib/python3.12/socket.py:720, in SocketIO.
 ↪readinto(self, b)
    718 while True:
    719     try:
--> 720         return self._sock.recv_into(b)
    721     except timeout:
```

```
    722         self._timeout_occurred = True

KeyboardInterrupt:
```

0.2 Requêtes sur le flux

[4]:
```
words = spark.read.table("array_words")
words.show()
```

```
+--------------------+--------------------+--------------------+----------+
|               value|           timestamp|              window|      word|
+--------------------+--------------------+--------------------+----------+
|L'éléphant détest…|2024-10-04 10:07:…|{2024-10-04 10:07…|L'éléphant|
|L'éléphant détest…|2024-10-04 10:07:…|{2024-10-04 10:07…|   déteste|
|L'éléphant détest…|2024-10-04 10:07:…|{2024-10-04 10:07…|        la|
|L'éléphant détest…|2024-10-04 10:07:…|{2024-10-04 10:07…|   musique|
|L'étudiant constr…|2024-10-04 10:10:…|{2024-10-04 10:10…|L'étudiant|
|L'étudiant constr…|2024-10-04 10:10:…|{2024-10-04 10:10…| construit|
|L'étudiant constr…|2024-10-04 10:10:…|{2024-10-04 10:10…|        la|
|L'étudiant constr…|2024-10-04 10:10:…|{2024-10-04 10:10…|   musique|
|L'étudiant constr…|2024-10-04 10:06:…|{2024-10-04 10:06…|L'étudiant|
|L'étudiant constr…|2024-10-04 10:06:…|{2024-10-04 10:06…| construit|
|L'étudiant constr…|2024-10-04 10:06:…|{2024-10-04 10:06…|        la|
|L'étudiant constr…|2024-10-04 10:06:…|{2024-10-04 10:06…|   musique|
|Le professeur con…|2024-10-04 10:10:…|{2024-10-04 10:10…|        Le|
|Le professeur con…|2024-10-04 10:10:…|{2024-10-04 10:10…|professeur|
|Le professeur con…|2024-10-04 10:10:…|{2024-10-04 10:10…| construit|
|Le professeur con…|2024-10-04 10:10:…|{2024-10-04 10:10…|        le|
|Le professeur con…|2024-10-04 10:10:…|{2024-10-04 10:10…|    jardin|
|Le professeur con…|2024-10-04 10:06:…|{2024-10-04 10:06…|        Le|
|Le professeur con…|2024-10-04 10:06:…|{2024-10-04 10:06…|professeur|
|Le professeur con…|2024-10-04 10:06:…|{2024-10-04 10:06…| construit|
+--------------------+--------------------+--------------------+----------+
only showing top 20 rows
```

[11]:
```
words.filter(
    (col("window.start") >= F.lit("2024-10-04 10:05:00")) &
    (col("window.end") <= F.lit("2024-10-04 10:07:00"))
).show()
```

```
+--------------------+--------------------+--------------------+----------+
|               value|           timestamp|              window|      word|
+--------------------+--------------------+--------------------+----------+
|L'étudiant constr…|2024-10-04 10:06:…|{2024-10-04 10:06…|L'étudiant|
|L'étudiant constr…|2024-10-04 10:06:…|{2024-10-04 10:06…| construit|
|L'étudiant constr…|2024-10-04 10:06:…|{2024-10-04 10:06…|        la|
|L'étudiant constr…|2024-10-04 10:06:…|{2024-10-04 10:06…|   musique|
+--------------------+--------------------+--------------------+----------+
```

```
|Le professeur con…|2024-10-04 10:06:…|{2024-10-04 10:06…|        Le|
|Le professeur con…|2024-10-04 10:06:…|{2024-10-04 10:06…|professeur|
|Le professeur con…|2024-10-04 10:06:…|{2024-10-04 10:06…| construit|
|Le professeur con…|2024-10-04 10:06:…|{2024-10-04 10:06…|        le|
|Le professeur con…|2024-10-04 10:06:…|{2024-10-04 10:06…|    jardin|
|Le professeur con…|2024-10-04 10:05:…|{2024-10-04 10:05…|        Le|
|Le professeur con…|2024-10-04 10:05:…|{2024-10-04 10:05…|professeur|
|Le professeur con…|2024-10-04 10:05:…|{2024-10-04 10:05…| construit|
|Le professeur con…|2024-10-04 10:05:…|{2024-10-04 10:05…|        le|
|Le professeur con…|2024-10-04 10:05:…|{2024-10-04 10:05…|    jardin|
|Le professeur con…|2024-10-04 10:06:…|{2024-10-04 10:06…|        Le|
|Le professeur con…|2024-10-04 10:06:…|{2024-10-04 10:06…|professeur|
|Le professeur con…|2024-10-04 10:06:…|{2024-10-04 10:06…| construit|
|Le professeur con…|2024-10-04 10:06:…|{2024-10-04 10:06…|        le|
|Le professeur con…|2024-10-04 10:06:…|{2024-10-04 10:06…|    ballon|
|L'éléphant détrui…|2024-10-04 10:06:…|{2024-10-04 10:06…|L'éléphant|
+--------------------+--------------------+--------------------+----------+
only showing top 20 rows
```

[15]:
```
words.filter(
    (col("window.start") >= F.lit("2024-10-04 10:05:00")) &
    (col("window.end") <= F.lit("2024-10-04 10:07:00"))).groupBy("word").
 ↪count().show()
```

```
+----------+-----+
|      word|count|
+----------+-----+
|        Le|   21|
|    ballon|    7|
|    jardin|    6|
|   détruit|    4|
|professeur|    7|
|   fromage|    6|
| construit|    8|
|        le|   38|
|L'étudiant|    8|
|   musique|    5|
|        la|   12|
|L'éléphant|    7|
|      film|    8|
|   cherche|    6|
|     livre|    6|
|   déteste|    5|
|  enseigne|    3|
|   apprend|    6|
|     mange|    6|
|     pomme|    4|
```

```
+----------+-----+
only showing top 20 rows
```