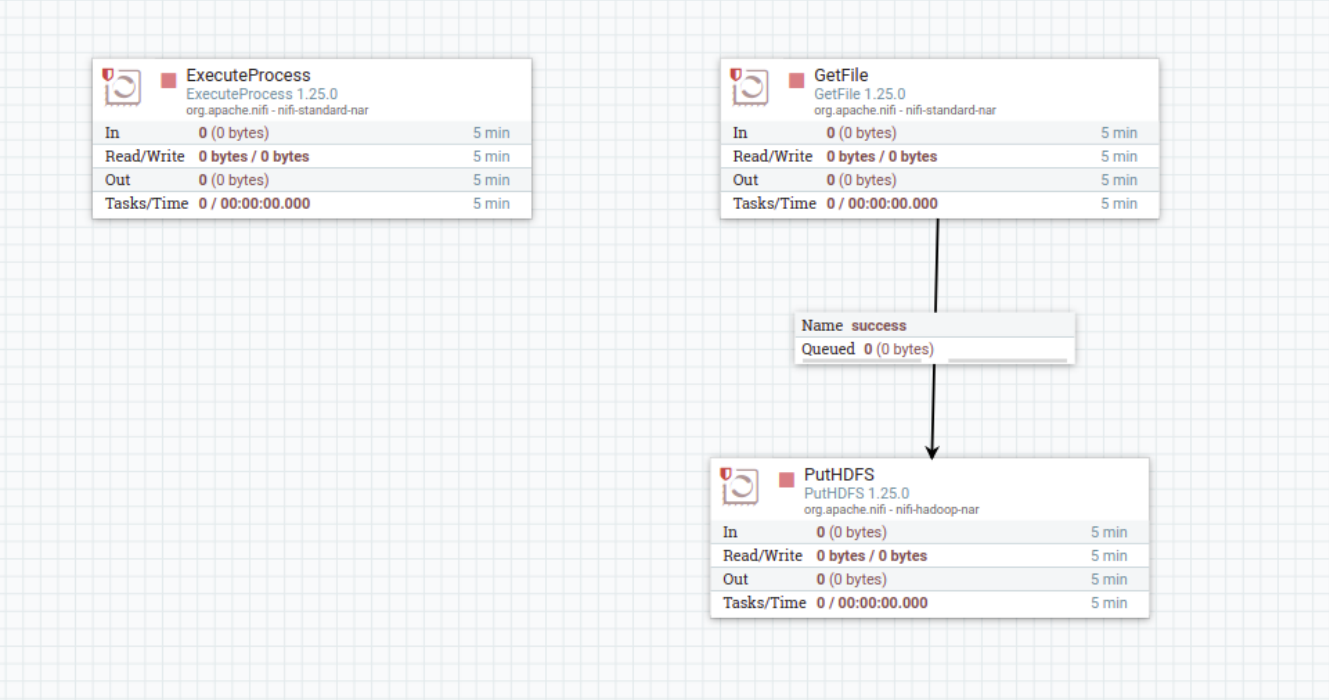
DSC 650 Weeks 11/12 – Final
Data Pipeline – NBA CSV Data
Xhoi Shyti
Sunday March 2, 2025

Data Pipeline Overview and Walkthrough

This report presents an overview and discussion of the data pipeline designed to ingest, move, and query NBA data originating from https://www.nba.com/stats. The initial steps in the pipeline are handled using Nifi – consider the below screenshot showing the Nifi processor group.



The first step in the pipeline is the **ExecuteProcessor** that executes a python script which makes several API calls to https://www.nba.com/stats, each corresponding to a particular NBA season.

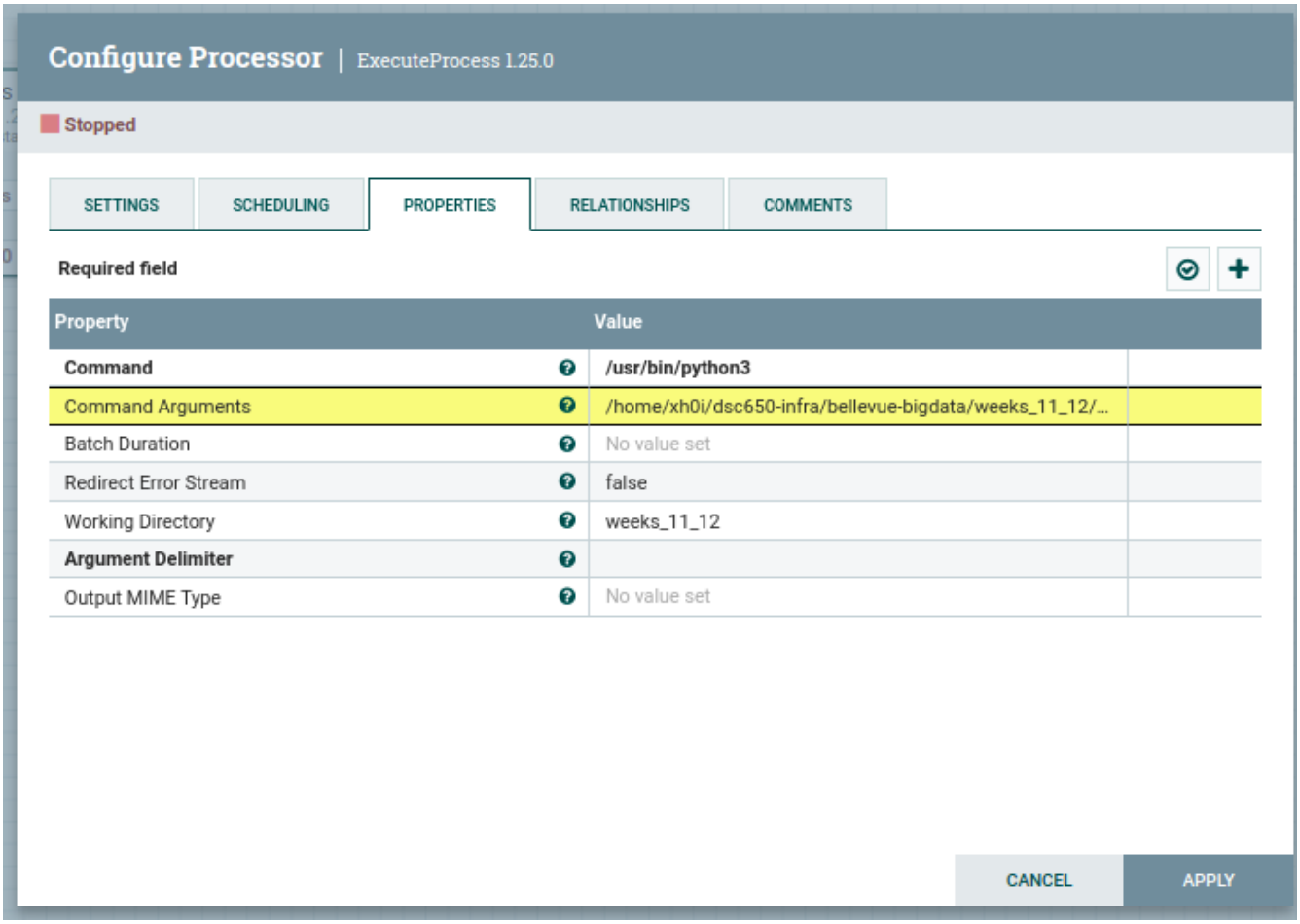
```
for i in range(1946, 2024):
    season_id = '{}-{}'.format(str(i), str(i+1)[-2:])

    url = 'https://stats.nba.com/stats/leaguegamelog?Counter=1000&DateFrom=&DateTo=&Direction=DESC&LeagueID=00&PlayerOrTeam=T&Season='+season_id+'&SeasonType=Regular%20Season&Sorter=DATE'
    headers  = {
        'Connection': 'keep-alive',
        'Accept': 'application/json, text/plain, */*',
        'x-nba-stats-token': 'true',
        'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/79.0.3945.130 Safari/537.36',
        'x-nba-stats-origin': 'stats',
        'Sec-Fetch-Site': 'same-origin',
        'Sec-Fetch-Mode': 'cors',
        'Referer': 'https://stats.nba.com/',
        'Accept-Encoding': 'gzip, deflate, br',
        'Accept-Language': 'en-US,en;q=0.9',
    }

    response = requests.get(url=url, headers=headers).json()
```

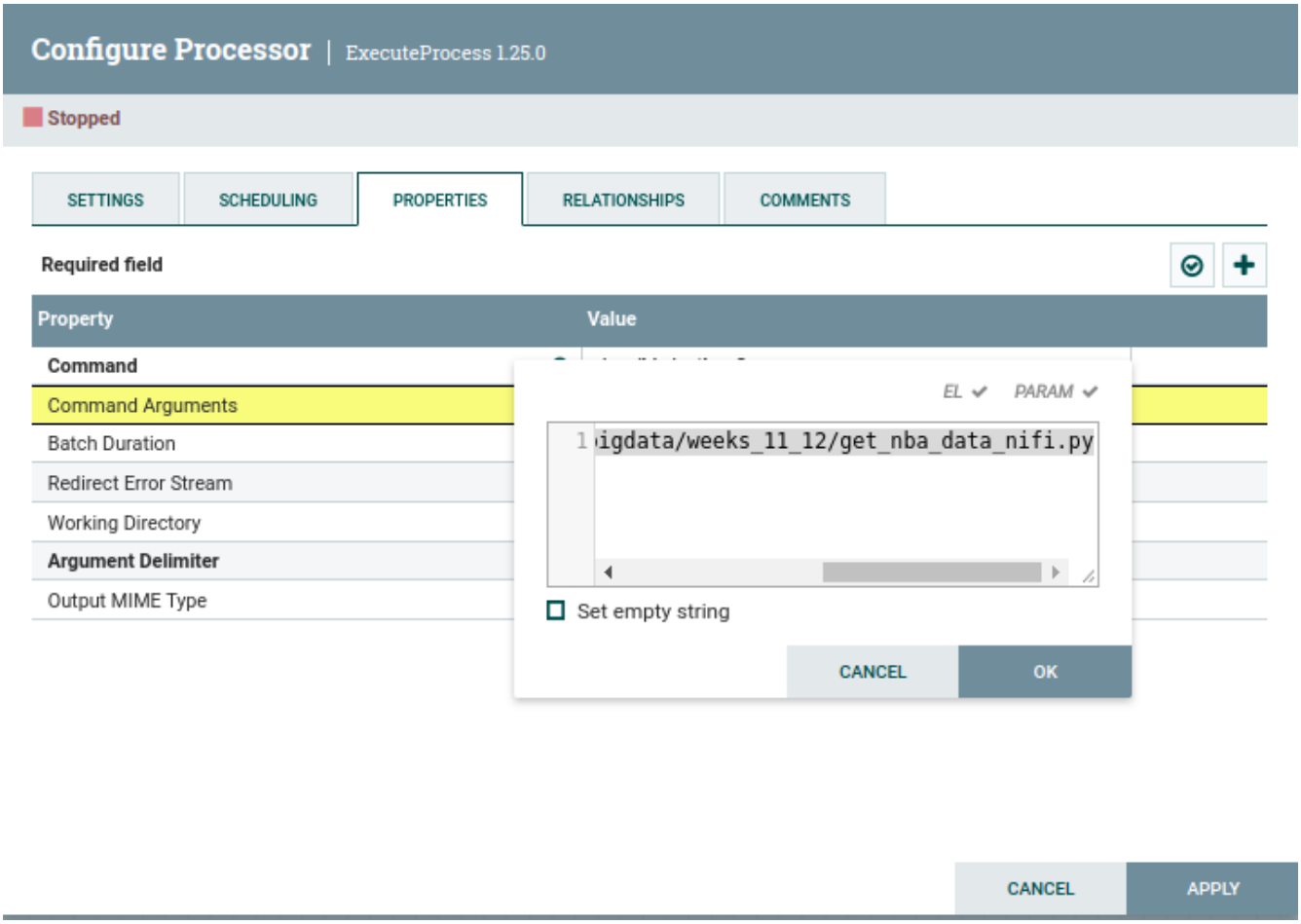
The loop iterates over valid NBA seasons, and a unique request URL is generated given the season. Using the headers listed, the python request library method ‘requests.get()’ calls the API and converts the response to JSON.

Then, additional processing logic converts the JSON to CSV format and outputs this. The result is a unique data file for each nba season. Finally, the script accumulates all of the separate CSV files into one single CSV. The below screenshot shows the details of the processing required.

```
        games = response['resultSets'][0]['rowSet']

        header = response['resultSets'][0]['headers']
        data_file_name = '{}.csv'.format(season_id)
        data_file = open(data_file_name, 'w')

        csv_writer = csv.writer(data_file)

        count = 0

        for game in games:
            if count == 0:

                header = header
                csv_writer.writerow(header)
                count += 1

            csv_writer.writerow(game)

        data_file.close()

    csv_folder = '/home/xh0i/dsc650-infra/bellevue-bigdata/nifi/nifi-1.25.0/weeks_11_12'

    combined_data = pd.DataFrame()

    for filename in os.listdir(csv_folder):
        if filename.endswith('.csv'):
            filepath = os.path.join(csv_folder, filename)
            df = pd.read_csv(filepath)
            combined_data = pd.concat([combined_data, df], ignore_index=True)

    output_file = '/home/xh0i/dsc650-infra/bellevue-bigdata/nifi/nifi-1.25.0/weeks_11_12/all_games.csv'
    combined_data.to_csv(output_file, index=False)

main()
```

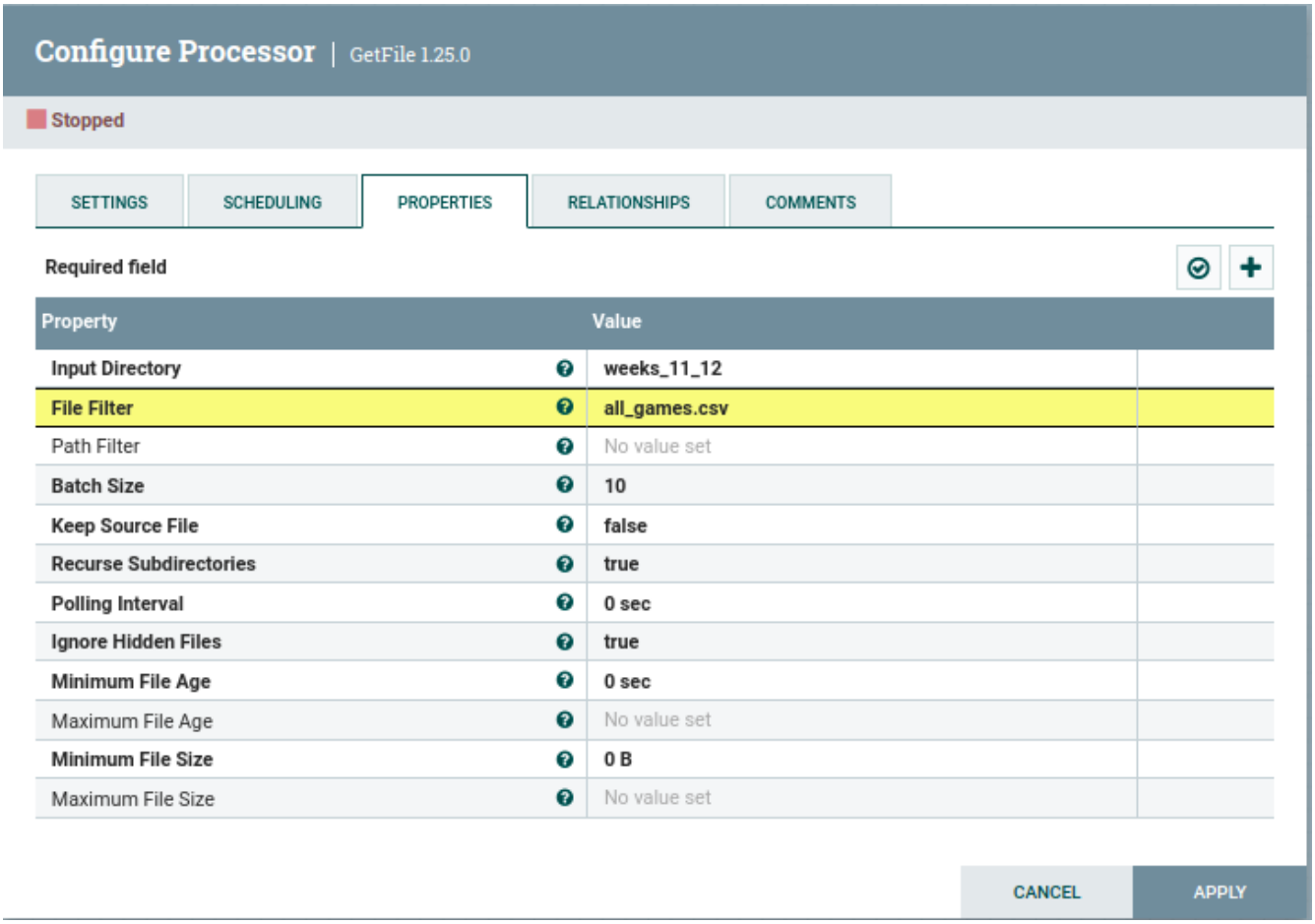
Notice that the output path of this is the dsc650-infra remote instance. Since Nifi is to execute this script, it must output to the remote instance itself.


Configure Processor | ExecuteProcess 1.25.0
Stopped
SETTINGS
SCHEDULING
PROPERTIES
RELATIONSHIPS
COMMENTS
Required field
Property
Value
Command
/usr/bin/python3
Command Arguments
/home/xh0i/dsc650-infra/bellevue-bigdata/weeks_11_12/...
Batch Duration
No value set
Redirect Error Stream
false
Working Directory
weeks_11_12
Argument Delimiter
Output MIME Type
No value set
CANCEL
APPLY



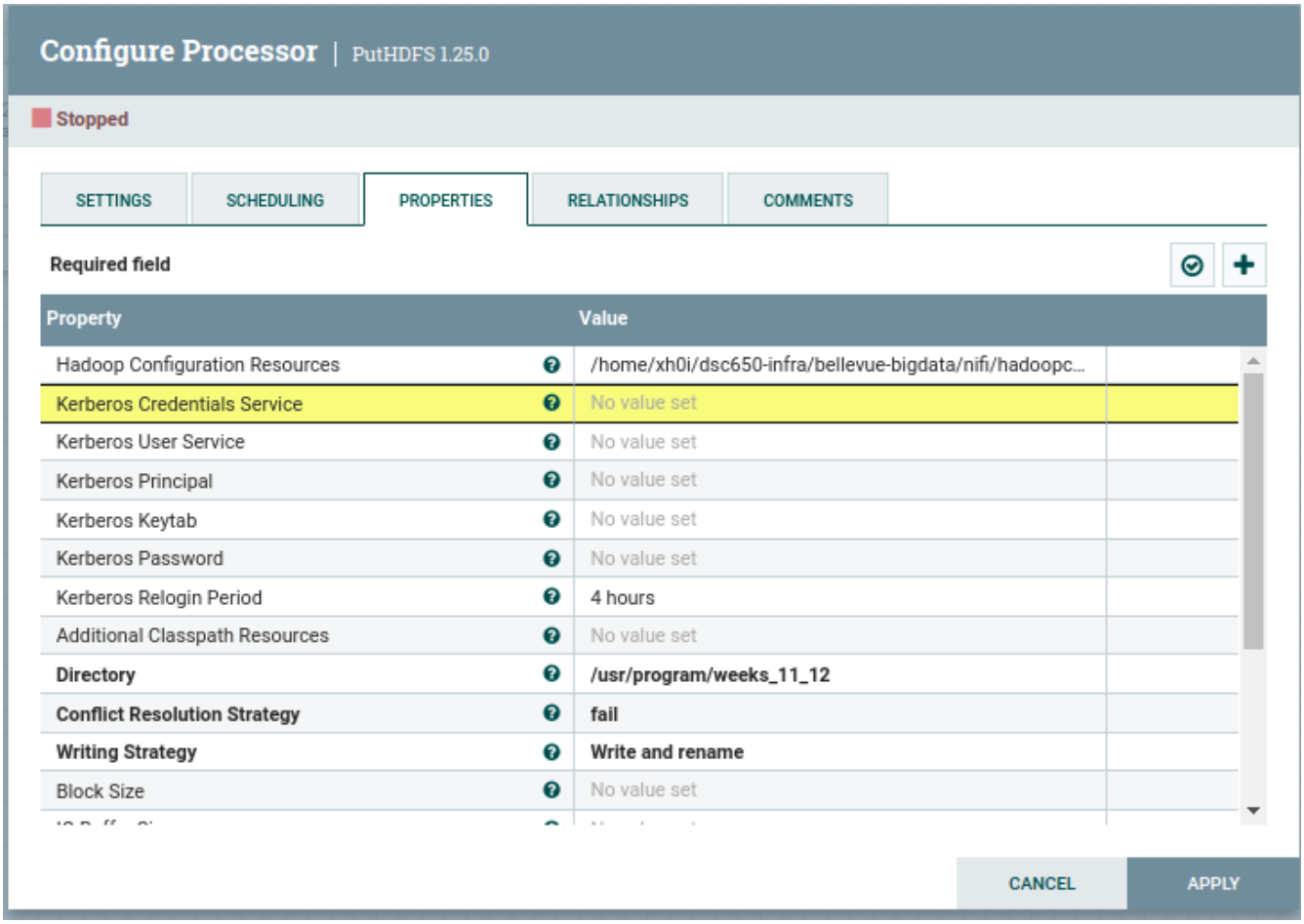
The properties were configured to enable this processor to execute the python script and output to weeks_11_12 directory.

The next step in the Nifi flow is the GetFile processor. This processor simply reads the CSV output and moves it.



Notice that the input directory matches the directory configured in the Execute Processor. Without this, the reader will not find the matching filter: “all_games.csv”.

Next, the PutHDFS processor takes the data moved by GetFile and transfers it to HDFS.



For that configuration, the Hadoop Configuration Resources must match that which is given in the nifi config itself in the remote instance. The directory points to the hdfs directory where the data is supposed to be transfered.

The below screenshot shows a successful search query using pyspark on hdfs.

```
bash-5.0# hdfs dfs -ls /usr/program/weeks_11_12
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/program/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/program/tez/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/program/hive/lib/log4j-slf4j-impl-2.10.0.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
2025-03-02 20:56:05,432 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 1 items
-rw-r--r--   3 xh0i supergroup   20245775 2025-03-02 20:55 /usr/program/weeks_11_12/all_games.csv
bash-5.0# pyspark
Python 3.7.10 (default, Mar  2 2021, 09:06:08)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/program/spark/jars/slf4j-log4j12-1.7.30.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/program/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
0    [main] WARN  org.apache.hadoop.util.NativeCodeLoader  - Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
922  [Thread-4] WARN  org.apache.hadoop.hive.conf.HiveConf  - HiveConf of name hive.strict.managed.tables does not exist
922  [Thread-4] WARN  org.apache.hadoop.hive.conf.HiveConf  - HiveConf of name hive.create.as.insert.only does not exist
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 3.0.0
      /_/

Using Python version 3.7.10 (default, Mar  2 2021 09:06:08)
SparkSession available as 'spark'.
>>> spark = SparkSession.builder \
...     .appName("NBA Data to Hive") \
...     .enableHiveSupport() \
...     .getOrCreate()
>>> grades_df = spark.read.option("header", True) \
...     .csv("/usr/program/weeks_11_12/all_games.csv")
>>> grades_df.write.mode("overwrite").saveAsTable("default.nbadata")
102871 [Thread-4] WARN  org.apache.hadoop.hive.conf.HiveConf  - HiveConf of name hive.strict.managed.tables does not exist
102872 [Thread-4] WARN  org.apache.hadoop.hive.conf.HiveConf  - HiveConf of name hive.create.as.insert.only does not exist
102912 [Thread-4] WARN  org.apache.spark.sql.hive.client.HiveClientImpl  - Detected HiveConf hive.execution.engine is 'tez' and will be reset to 'mr' to disable useless hive logic
104694 [Thread-4] WARN  org.apache.spark.sql.catalyst.util.package  - Truncated the string representation of a plan since it was too large. This behavior can be adjusted by setting 'spark.sql.debug.maxToStringFields'.
114780 [Thread-4] WARN  org.apache.hadoop.hive.ql.session.SessionState  - METASTORE_FILTER_HOOK will be ignored, since hive.security.authorization.manager is set to instance of HiveAuthorizerFactory.
>>> spark.sql("SELECT * FROM default.nbadata limit 10").show()
+---------+----------+-----------------+--------------------+--------+----------+-----------+---+---+----+----+------+----+----+-------+----+----+------+----+----+----+----+----+----+----+----+---+----------+---------------+
|SEASON_ID|   TEAM_ID|TEAM_ABBREVIATION|           TEAM_NAME| GAME_ID| GAME_DATE|    MATCHUP| WL|MIN| FGM| FGA|FG_PCT|FG3M|FG3A|FG3_PCT| FTM| FTA|FT_PCT|OREB|DREB| REB| AST| STL| BLK| TOV|  PF|PTS|PLUS_MINUS|VIDEO_AVAILABLE|
+---------+----------+-----------------+--------------------+--------+----------+-----------+---+---+----+----+------+----+----+-------+----+----+------+----+----+----+----+----+----+----+----+---+----------+---------------+
|    21989|1610612753|              ORL|       Orlando Magic|28901098|1990-04-22|  ORL @ NJN|  W|240|39.0|83.0|  0.47| 1.0| 5.0|    0.2|31.0|43.0| 0.721|11.0|31.0|42.0|15.0|10.0| 5.0|19.0|24.0|110|         8|              0|
|    21989|1610612760|              SEA| Seattle SuperSonics|28901102|1990-04-22|  SEA @ GOS|  L|240|45.0|96.0| 0.469| 4.0|14.0|  0.286|28.0|37.0| 0.757|26.0|30.0|56.0|29.0| 7.0| 2.0|22.0|31.0|122|        -2|              0|
|    21989|1610612743|              DEN|      Denver Nuggets|28901103|1990-04-22|DEN vs. MIN|  W|240|39.0|83.0|  0.47| 3.0| 9.0|  0.333|34.0|40.0|  0.85|18.0|30.0|48.0|20.0| 8.0| 7.0|13.0|23.0|115|         7|              0|
|    21989|1610612738|              BOS|      Boston Celtics|28901097|1990-04-22|  BOS @ PHL|  W|240|50.0|98.0|  0.51| 0.0| 1.0|    0.0|18.0|21.0| 0.857|20.0|32.0|52.0|29.0|13.0| 6.0|11.0|23.0|118|        20|              0|
|    21989|1610612745|              HOU|     Houston Rockets|28901107|1990-04-22|HOU vs. UTH|  W|240|39.0|94.0| 0.415| 2.0| 8.0|   0.25|20.0|34.0| 0.588|21.0|36.0|57.0|20.0|13.0| 8.0|16.0|22.0|100|        12|              0|
|    21989|1610612752|              NYK|     New York Knicks|28901106|1990-04-22|  NYK @ CLE|  L|240|43.0|81.0| 0.531| 0.0| 7.0|    0.0|13.0|19.0| 0.684| 9.0|26.0|35.0|31.0| 9.0| 4.0|17.0|18.0| 99|       -16|              0|
|    21989|1610612739|              CLE| Cleveland Cavaliers|28901106|1990-04-22|CLE vs. NYK|  W|240|51.0|96.0| 0.531| 3.0| 7.0|  0.429|10.0|14.0| 0.714|21.0|26.0|47.0|37.0|10.0| 7.0|15.0|16.0|115|        16|              0|
|    21989|1610612756|              PHX|        Phoenix Suns|28901101|1990-04-22|  PHX @ SAN|  L|240|38.0|90.0| 0.422| 1.0| 5.0|    0.2|16.0|19.0| 0.842|12.0|32.0|44.0|19.0| 7.0| 4.0|14.0|23.0| 93|       -15|              0|
|    21989|1610612744|              GOS|Golden State Warr...|28901102|1990-04-22|GOS vs. SEA|  W|240|44.0|81.0| 0.543| 3.0| 8.0|  0.375|33.0|39.0| 0.846| 9.0|24.0|33.0|26.0|12.0| 7.0|18.0|25.0|124|         2|              0|
|    21989|1610612762|              UTH|           Utah Jazz|28901107|1990-04-22|  UTH @ HOU|  L|240|33.0|75.0|  0.44| 5.0|11.0|  0.455|17.0|23.0| 0.739| 5.0|35.0|40.0|26.0| 5.0|11.0|22.0|29.0| 88|       -12|              0|
+---------+----------+-----------------+--------------------+--------+----------+-----------+---+---+----+----+------+----+----+-------+----+----+------+----+----+----+----+----+----+----+----+---+----------+---------------+

>>>
```

Challenges Faced

Finding the correct processor to execute the python script took some research, and the pip tool had to be installed on the remote instance, as well as the pandas library after that.

There was some trial and error to cofigure the GetFile processor, as it was not immediately obvious that the directories should match.

There was additional trial and error the PutHDFS processor, since the first attempts were, once again, pointing to the wrong directory.

Several times the remote instance itself had to be restarted since the shell would freeze – likely due to the fact that Nifi was kept running and that, by itself, is very resource intensive for the remote instance.

Conclusion

This data pipeline effectively integrates Apache NiFi, HDFS, Hive, and Spark to facilitate seamless data ingestion, storage, and analysis. By leveraging NiFi, data is efficiently ingested from https://www.nba.com/stats and loaded into HDFS. The integration with Hive enables structured querying and metadata management, providing a structured framework for data organization. Finally, Apache Spark is used to execute high-performance queries and transformations, allowing for advanced analytics and data processing.

This end-to-end workflow ensures an automated, scalable, and efficient data processing pipeline, making it well-suited for big data applications. The combination of these technologies enhances data

accessibility, performance, and analytical capabilities, supporting informed decision-making and real-time insights.