Evaluation Practical Work

Big Data Infrastructure & Cloud Computing

Recommended IDE: PyCharm Community Edition
Official documentation for Spark: https://spark.apache.org/docs/latest/

Deliverable:
- Provide a report showing the results for every steps, using screenshots proving that you could complete the step.
- The quality of presentation for the report will count in the score. It can be simple but should look clean and professional, with good spelling and grammar.

1. Create a virtual environment in PyCharm and set it up to use Python 3.8 (or another version 3.x already installed on your computer, no need to install 3.8 if you don't have it yet).

Tips: https://www.jetbrains.com/help/pycharm/creating-virtual-environment.html

2. Write a script in Python shakespeare_words.py which generates a list of the words contained in the complete works of Shakespeare and output it to a file. Name this file shakespeare_data.txt.

The complete works of Shakespeare are available here (the header and other texts which are not related to the original text should be removed):
https://ocw.mit.edu/ans7870/6/6.006/s08/lecturenotes/files/t8.shakespeare.txt

Tips:
- https://stackoverflow.com/a/6181784/9664780
- Output one word per line (like in the previous practical work)

3. Install PySpark using the command line:

```
$pip install pyspark
```

4. Test the installation by running the PySpark shell:

```
$pyspark
```

5. Load your shakespeare_data.txt file in the shell and count the number of words in the file.

Tips: create a DataFrame and then use the count() method.
https://spark.apache.org/docs/latest/quick-start.html#basics

6. Exit the shell:

```
>>> exit()
```

7. Create a word_count.py file and do the same thing like with the shell (reading the text file and displaying the number of words). For that, you need to create an instance of spark in your script using a SparkSession:

```
spark = (SparkSession.builder.appName('WordCountApp').getOrCreate())
```

8. Then add spark code to your script in order to show:

- The first three values in the text file
- The 10 longest words, showing their length
- The 10 words having the highest number of occurrences, with their number of occurrences

9. Run your script locally and observe the results.

10. Create an Amazon EMR cluster (keep default parameters).

11. Create a S3 bucket and upload your data to the bucket.

12. Open the Spark History Server user interface.

13. Run your script on the cluster using the graphical user interface. Observe the jobs being run in the Spark History Server.

Tips:
https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-gs.html

14. Observe the final state in Spark History Server.

15. Propose and test different optimizations to improve the performances. Show the performance gains attained with each optimization tested.

Tips:
https://mageswaran1989.medium.com/spark-optimizations-for-advanced-users-spark-cheat-sheet-d74464618c20
Or google something like “spark performance tuning”.