

🌸 76. SPARK 설치

SPARK

=> hive 의 단점을 개선하기 위해서 나온게 스파크
=> hive 의 단점이 disk에서 데이터를 처리하기 때문에 속도가 느리다.
=> 스파크는 메모리에서 데이터를 처리해서 속도가 아주 빠르다. 요즘에 하드웨어 성능이 좋아지고 메모리 가격이 이전에 비해서 저렴해져서 스파크 (spark) 사용이 대중화되고 있다.

SPARK의 특징

=> 메모리 기반의 데이터 처리로 속도가 아주 빠르다
=> 하둡의 hdfs 를 이용할 수도 있고 또는 단독으로 사용이 가능하다.
=> 파이썬과 연동을 해서 다양한 파이썬의 패키지들을 사용할 수 있다.

SPARK 설치

1. oracle 의 홈디렉토리로 이동한다.

cd

2. 설치 파일을 다운로드 받는다.

wget <https://archive.apache.org/dist/spark/spark-2.0.2/spark-2.0.2-bin-hadoop2.7.tgz>

3. 압축을 푼다.

tar xvzf spark-2.0.2-bin-hadoop2.7.tgz

4. 압축을 풀고 생긴 디렉토리의 이름을 spark로 변경한다.

mv spark-2.0.2-bin-hadoop2.7 spark

5. spark 파일 안에 있는 bin 폴더에 spark-shell 이 있는지 확인한다.

cd spark

ls

```
[oracle@centos ~]$ mv spark-2.0.2-bin-hadoop2.7 spark
[oracle@centos ~]$ cd spark
[oracle@centos spark]$ ls
LICENSE NOTICE R README.md RELEASE bin conf data examples jars licenses python sbin yarn
[oracle@centos spark]$ cd bin
[oracle@centos bin]$ ls
beeline pyspark run-example.cmd spark-shell spark-submit sparkR.cmd
beeline.cmd pyspark.cmd spark-class spark-shell2.cmd spark-submit2.cmd sparkR2.cmd
load-spark-env.cmd pyspark2.cmd spark-class2.cmd spark-shell2.cmd spark-submit2.cmd
load-spark-env.sh run-example spark-class2.cmd spark-sql sparkR
```

6. spark을 실행해본다.

./spark-shell

```
[oracle@centos bin]$ ./spark-shell
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel).
24/01/26 14:02:05 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-
java classes where applicable
24/01/26 14:02:07 WARN SparkContext: Use an existing SparkContext, some configuration may not take effect.
Spark context Web UI available at http://192.168.122.1:4040
Spark context available as 'sc' (master = local[*], app id = local-1706245326847).
Spark session available as 'spark'.
Welcome to

  ____  __
 / ___/  / /
/ /   /  / /
/ /___/  / /
/_____/  / /
         /_/

version 2.0.2

Using Scala version 2.11.8 (Java HotSpot(TM) Client VM, Java 1.7.0_60)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

7. oracle 계정으로 돌아와서 .bash_profile를 열어서 아래의 export 문을 입력한다. (spark 단축키)

```
vi .bash_profile
```

```
-----
export PATH=$PATH:/home/oracle/spark/bin:$PATH
-----
source .bash_profile
```

8. spark에서 사용할 employee.txt를 생성한다.

```
vi employee.txt
```

```
-----
1201,satish,25
1202,krishna,28
1203,amith,39
1204,javed,23
1205,prudvi,23
-----
```

9. 테이블 생성 전에 아래의 명령어를 실행한다.

```
(scala)
```

```
val sqlContext = new org.apache.spark.sql.hive.HiveContext(sc)
```

10. OS에 저장한 employee.txt 데이터를 저장할 테이블을 생성한다.

```
sqlContext.sql("CREATE TABLE IF NOT EXISTS employee (id INT, name STRING, age INT)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n'")
```

11. os 의 employee.txt 파일을 employee 에 로드한다.

```
sqlContext.sql("LOAD DATA LOCAL INPATH 'employee.txt' INTO TABLE employee")
```

12. employee 테이블에서 id 와 name과 age 를 조회한다.

```
val result = sqlContext.sql("FROM employee SELECT id, name, age")
```

13. result 에 입력된 결과를 출력한다.

```
result.show()
```

14. result 라는 변수에 담지 않고 바로 sql 을 실행하고 싶다면 아래와 같이 수행하시오.

```
sql("select * from employee").show()
```

spark에서 create table 문장 실행 후 에러 발생 시 해결 방법
아래의 에러 발생 시,

```
Failed to start database 'metastore_db' with class loader
```

oracle 유저로 가서 아래의 해당 파일이 존재하는지 확인한다. 없다면, sparko를 다시 설치한다.

```
ls -l | grep metastore_db
```

```
rm metastore_db/dbex.lock
```