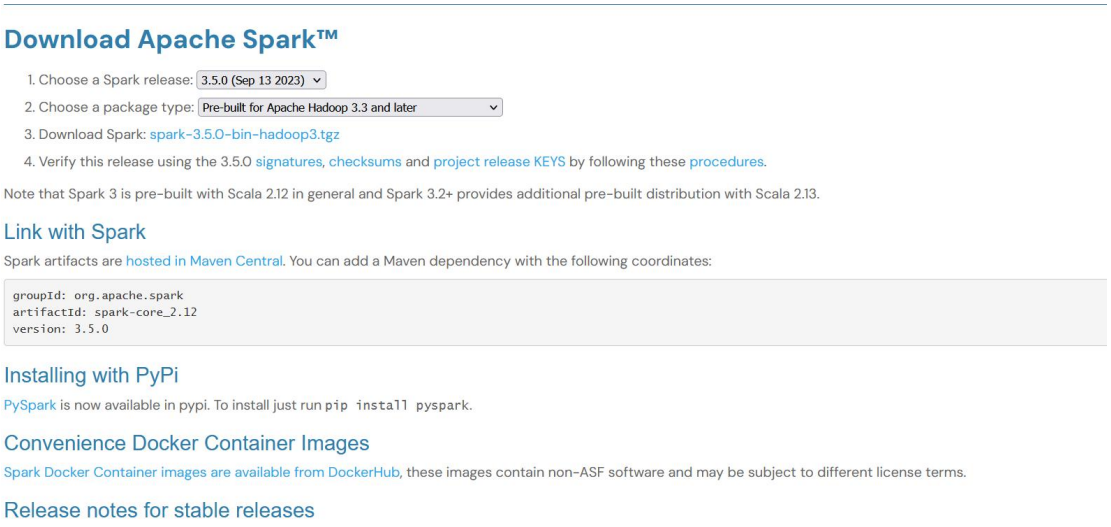
SPARK 实践

Spark 简介：

Apache Spark 是专门为大数据处理而设计的通用的计算引擎。spark 拥有 MapReduce 所具有的优点，但不同于 Map Reduce 的是 Job 中间输出结果可以缓存到内存中，从而不再需要读写 HDFS，减少磁盘数据交互，因此 Spark 能更好的适应机器学习和数据挖掘等需要迭代的算法。Spark 提供了 Spark RDD 、 Spark SQL 、 Spark Streaming 、 Spark MLlib 、 Spark GraphX 等技术组件，可以一站式地完成大数据领域的离线批处理、交互式查询、流式计算、机器学习、图计算等常见的任务。这就是 spark 一站式开发的特点。

Spark 安装

官网：https://spark.apache.org/downloads.html



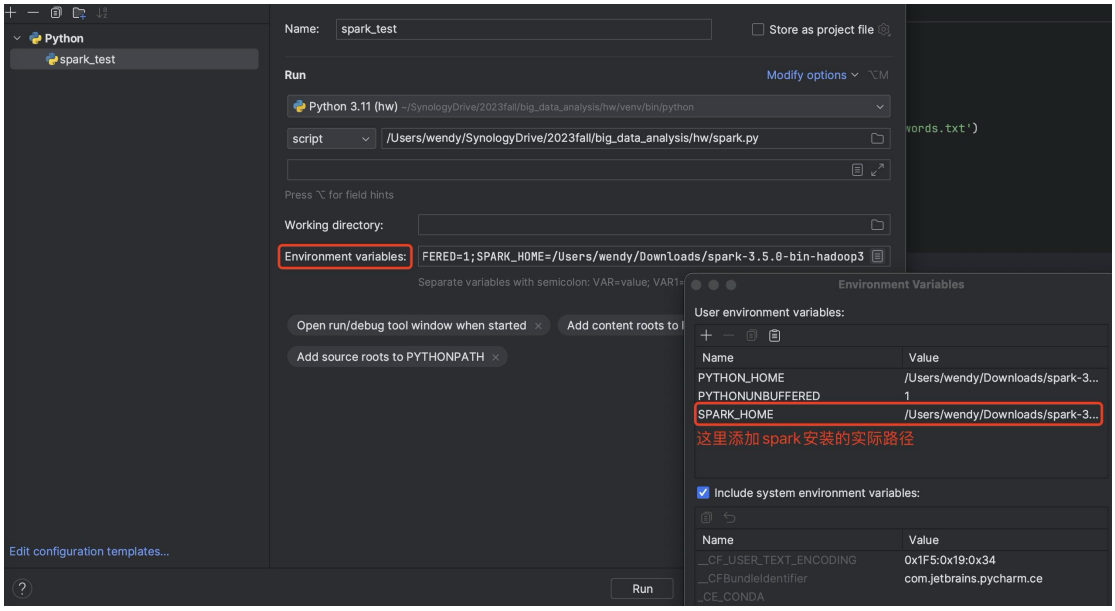
解压即可。在终端中输入 pySpark，如果安装成功会看到下图：

```
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 3.5.0
      /_/

Using Python version 3.11.5 (main, Sep 11 2023 08:31:25)
Spark context Web UI available at http://172.25.142.242:4040
Spark context available as 'sc' (master = local[*], app id = local-1701351226916
).
SparkSession available as 'spark'.
```

配置 Pycharm 中调用 pyspark 的加载包

1. 补充配置。PyCharm->Run->Edit Configurations，添加环境变量。

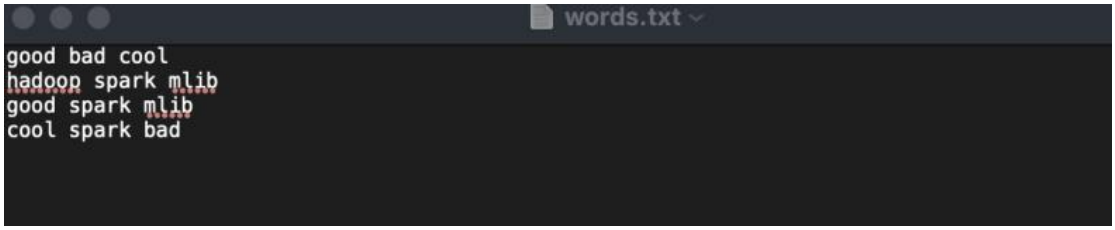


2. 初步运行

```python
from operator import add
from pyspark import SparkContext

if __name__ == "__main__":
    sc = SparkContext(appName="PythonWordCount")
    lines = sc.textFile('words.txt')
    counts = lines.flatMap(lambda x: x.split(' ')) \
                  .map(lambda x: (x,1)) \
                  .reduceByKey(add)
    output = counts.collect()
    for (word, count) in output:
        print("%s: %i" % (word, count))
    sc.stop()
```

Words.txt 文件



运行结果

Pyspark 使用语法

1. count 返回元素个数

```
from pyspark import SparkContext
sc = SparkContext("local", "count app")
words = sc.parallelize(
    ["scala",
     "java",
     "hadoop",
     "spark",
     "akka",
     "spark vs hadoop",
     "pyspark",
     "pyspark and spark"
     ])
counts = words.count()
print("Number of elements in RDD -> %i" % counts)
```

```
Number of elements in RDD -> 8
```

2. collect()返回所有元素，并转换为 python 数据类型

```
from pyspark import SparkContext
sc = SparkContext("local", "collect app")
words = sc.parallelize(
   ["scala",
    "java",
    "hadoop",
    "spark",
    "akka",
    "spark vs hadoop",
    "pyspark",
    "pyspark and spark"
    ])
coll = words.collect()
print("Elements in RDD -> %s" % coll)
```

```
Elements in RDD -> ['scala', 'java', 'hadoop', 'spark', 'akka', 'spark vs hadoop', 'pyspark', 'pyspark and spark']
```

3. filter()返回一个包含元素的新的 RDD,满足过滤器内部功能

```
from pyspark import SparkContext
sc = SparkContext("local", "Filter app")
words = sc.parallelize(
   ["scala",
    "java",
    "hadoop",
    "spark",
    "akka",
    "spark vs hadoop",
    "pyspark",
    "pyspark and spark"]
   )
words_filter = words.filter(lambda x: 'spark' in x)
filtered = words_filter.collect()
print("Filtered RDD -> %s" % (filtered))
```

```
Filtered RDD -> ['spark', 'spark vs hadoop', 'pyspark', 'pyspark and spark']
```

4. map(f,preservesPartitioning=False)，通过将该函数应用于 RDD 中的每个元素来返回新的 RDD。

```
from pyspark import SparkContext
sc = SparkContext("local", "Map app")
```

```
words = sc.parallelize(
    ["scala",
    "java",
    "hadoop",
    "spark",
    "akka",
    "spark vs hadoop",
    "pyspark",
    "pyspark and spark"]
    )
words_map = words.map(lambda x: (x, 1))
mapping = words_map.collect()
print("Key value pair -> %s" % (mapping))
```