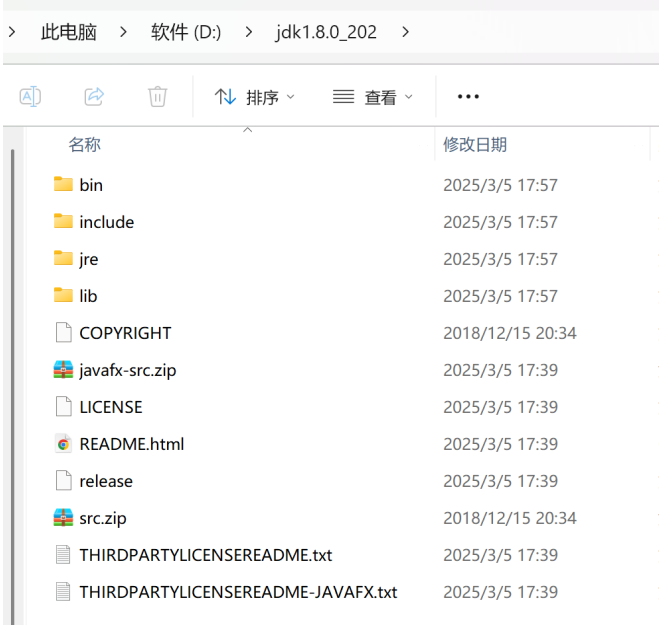
Pyspark开发环境配置

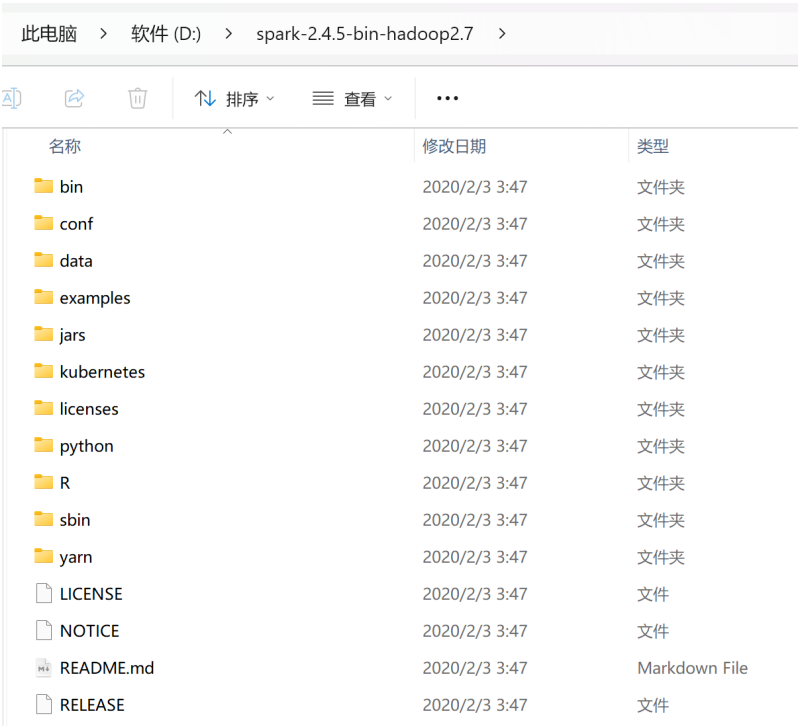
由于学习用的电脑配置及性能有限，原本在虚拟机中安装及部署的spark将在windows本机系统进行，各组件安装位置总结如下：

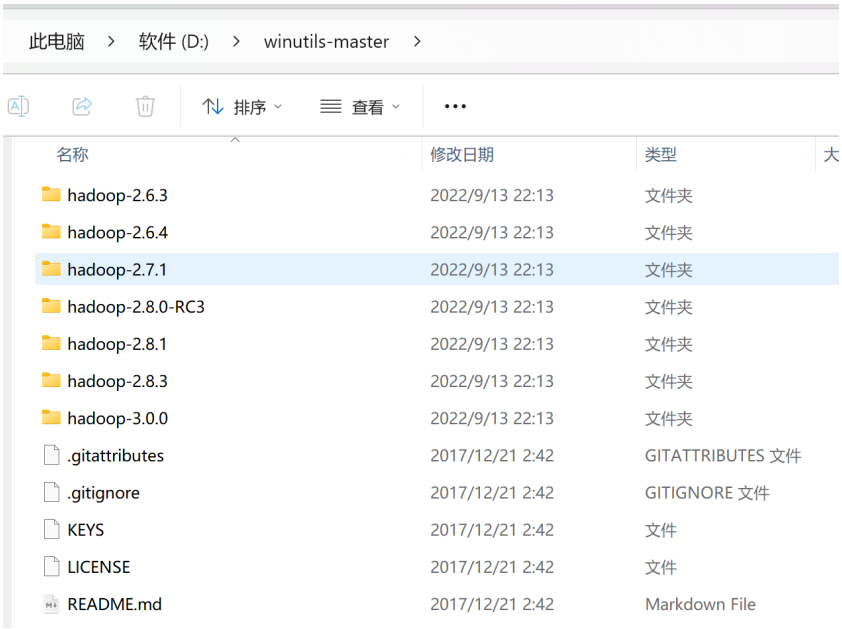
ubuntu虚拟机	windows本机
JDK	JDK
Hadoop	Python&Pycharm
Hive	Spark

一、windows环境配置

点击运行安装jdk（注意设置安装位置）、解压spark和winutils，建议三者放在同一个盘，勿直接使用默认C盘。最终得到三个文件夹。

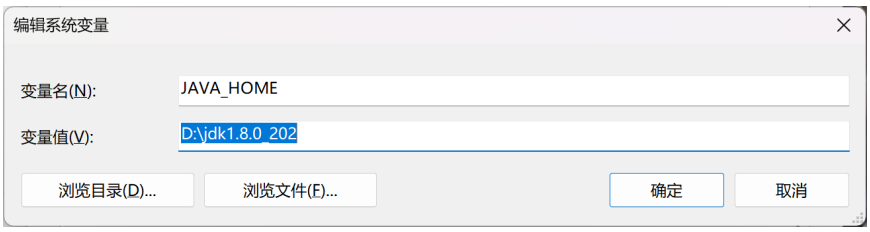


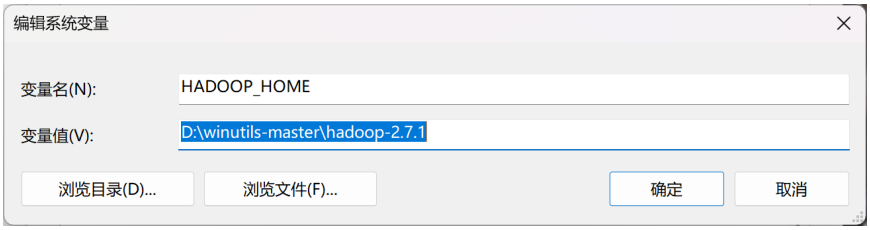


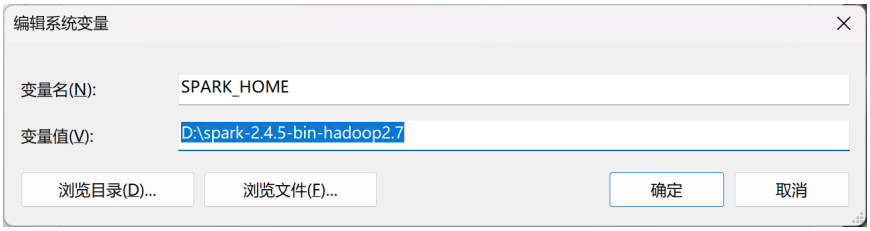


配置环境变量

在系统变量中添加 JAVA_HOME、HADOOP_HOME和SPARK_HOME 分别指向它们的安装根目录

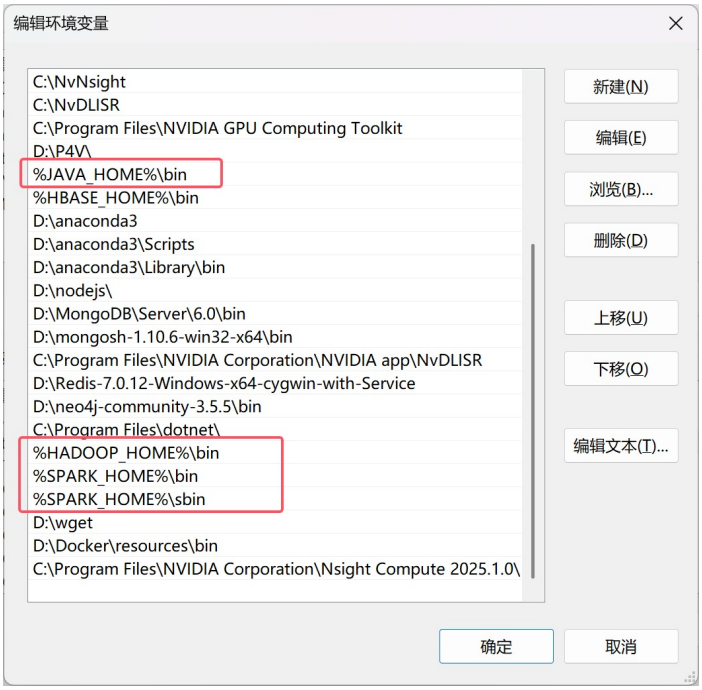






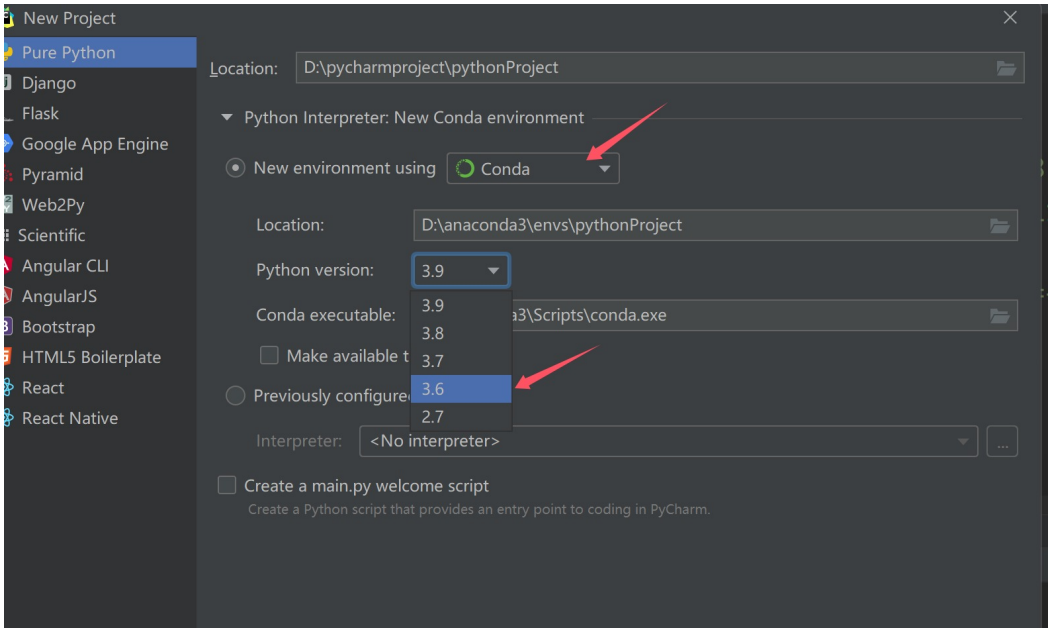
编辑Path路径，添加以下目录

```
%JAVA_HOME%\bin
%HADOOP_HOME%\bin
%SPARK_HOME%\bin
%SPARK_HOME%\sbin
```



二、Pycharm项目创建

创建pycharm项目时最好选择Conda环境，然后加载python3.6或接近的版本。



然后在Terminal窗口安装pyspark和py4j即可，需要指定对应版本号

```
pip install pyspark==2.4.5
pip install py4j==0.10.9
```

三、代码测试

创建py代码文件，编写基本入口代码，运行无任何报错则环境配置成功

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.getOrCreate()
```

```
D:\anaconda3\envs\TianM\python.exe D:/pycharmproject/TianM/1.py
Picked up JAVA_TOOL_OPTIONS: -Dfile.encoding=UTF-8
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). For SparkR, use setLogLevel(newLevel).

Process finished with exit code 0
```

红字提示非报错，可忽略。

四、Hive连接测试

打开虚拟机，启动hadoop，这里已提前建好了hive表并导入数据

```
hive> select * from behavior limit 100;
OK
492     254885  1       95lko70 2014    12      7       1
492     254885  1       95lko71 2014    12      7       2
492     254885  1       95lko75 2014    12      7       1
492     254885  1       95lko7r 2014    12      7       2
492     254885  1       95lkoro 2014    12      7       1
492     254885  1       95lkowd 2014    12      7       1
492     254885  1       95lkown 2014    12      7       1
492     254885  3       95lko71 2014    12      7       1
492     254885  4       95lkori 2014    12      7       1
492     2316002 1       95lko7v 2014    12      9       1
492     2316002 1       95lkora 2014    12      9       1
492     2316002 1       95lkowk 2014    12      9       1
492     3473697 1       95lko7u 2014    12      12      1
492     3473697 1       95lkowj 2014    12      12      1
492     6983065 1       95lko7s 2014    12      13      1
492     6983065 1       95lkorh 2014    12      13      1
```

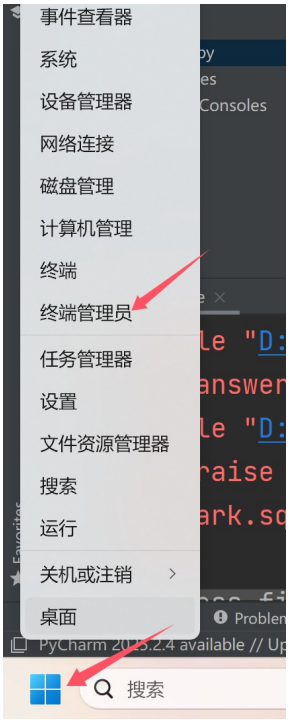
打开一个新的终端，输入以下命令启动Hive Metastore服务，以便外部可以访问hive元数据

```
nohup hive --service metastore
```

```
hadoop@sz:~/桌面$ nohup hive --service metastore
nohup: 忽略输入并把输出追加到 'nohup.out'
```

看到以上字样则保留该窗口不要关闭

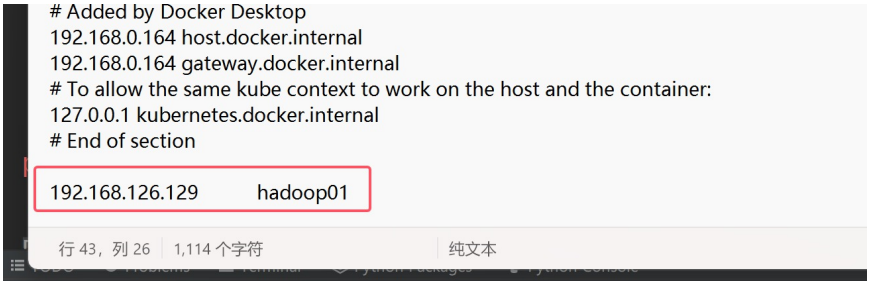
回到windows本机，修改hosts文件，该文件通常需要管理员权限才能修改。在开始按钮右键，打开终端管理员



进入hosts文件所在目录（以win11为例），用记事本打开

```
cd C:\Windows\System32\drivers\etc
notepad hosts
```

在末尾添加IP地址与名字的映射，地址为你的虚拟机主节点的IP地址



回到pyspark代码，创建入口时补充相关配置，有几处IP地址均改为你的虚拟机主节点IP地址

```
spark = SparkSession.builder \
    .config("spark.sql.warehouse.dir", "hdfs://你的虚拟机
IP:9000/user/hive/warehouse") \
    .config("hive.metastore.uris", "thrift://你的虚拟机IP:9083") \
    .config("spark.sql.catalogImplementation", "hive") \
    .config("spark.hadoop.hive.metastore.uris", "thrift://你的虚拟机IP:9083") \
    .enableHiveSupport() \
    .getOrCreate()
```

连接数据库，查表测试（根据自己的数据库和表名字修改）

```
spark.sql("use sz")
df_item = spark.sql("select * from item")
df_user = spark.sql("select * from behavior")
df_item.show()
df_user.show()
```

```
+---------+------------+-------------+
|  item_id|item_geohash|item_category|
+---------+------------+-------------+
|  item_id|item_geohash|         null|
|100002303|            |         3368|
|100003592|            |         7995|
|100006838|            |        12630|
|100008089|            |         7791|
```

```
+-------+-------+-------------+------------+----+-----+----+-----+
|user_id|item_id|behavior_type|user_geohash|year|month| day|count|
+-------+-------+-------------+------------+----+-----+----+-----+
|user_id|item_id|         null|user_geohash|null| null|null| null|
|    492| 254885|            1|     95lko70|2014|   12|   7|    1|
|    492| 254885|            1|     95lko71|2014|   12|   7|    2|
|    492| 254885|            1|     95lko75|2014|   12|   7|    1|
```

运行发现结果多了一行列名，这里修改sql语句进行简单过滤

```
spark.sql("use sz")
df_item = spark.sql("select * from item where item_id != 'item_id'")
df_user = spark.sql("select * from behavior where user_id != 'user_id'")
df_item.show()
df_user.show()
```

最终结果正常

```
+---------+------------+-------------+
|  item_id|item_geohash|item_category|
+---------+------------+-------------+
|100002303|            |         3368|
|100003592|            |         7995|
|100006838|            |        12630|
```

```
+-------+-------+-------------+------------+----+-----+---+-----+
|user_id|item_id|behavior_type|user_geohash|year|month|day|count|
+-------+-------+-------------+------------+----+-----+---+-----+
|    492| 254885|            1|     95lko70|2014|   12|  7|    1|
|    492| 254885|            1|     95lko71|2014|   12|  7|    2|
|    492| 254885|            1|     95lko75|2014|   12|  7|    1|
```

自行使用DataFrame操作进行不少于5个查询测试。