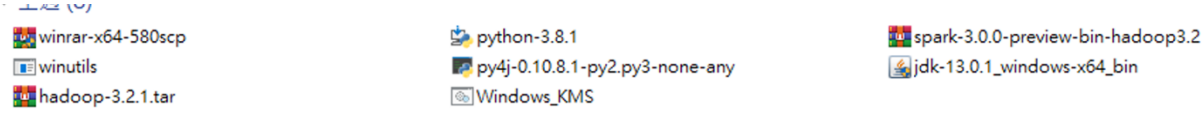
HW5

組員: 0853409 張舒婷 0853420 林若瑜 0853421 郭源芯

一、截圖與步驟描述

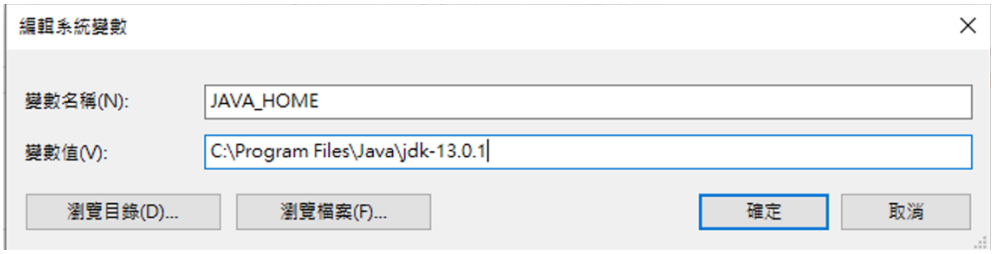
在 windows 環境底下安裝遇到了不少問題，因此也將問題和解決方法記錄起來

（一）下載相關檔案



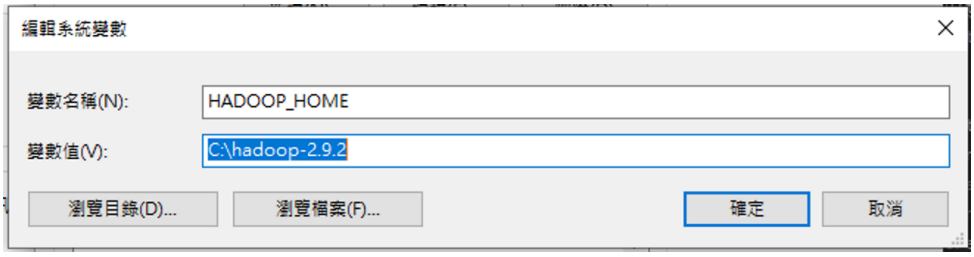
（二）安裝 JDK 並設置環境變量

JAVA 安裝目錄在 C:\Program Files\Java，需要新建系統變量 JAVA_HOME，值為 C:\Program Files\Java\jdk-13.0.1



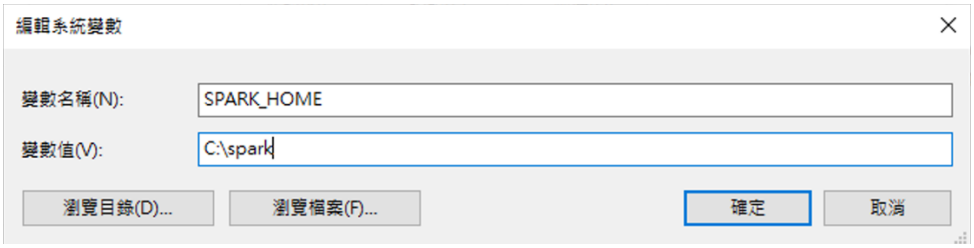
解壓 hadoop-2.9.2.tar.gz 到 C 盤根目錄

新建系統變量名 HADOOP_HOME， 變量值為 C:\hadoop-2.9.2



解壓 spark-3.0.0-bin-hadoop3.2.tgz 到 C 盤根目錄，spark-3.0.0 目錄重命名為 Spark。

新建系統變量名為 SPARK_HOME ,設置變量值為 C:\Spark

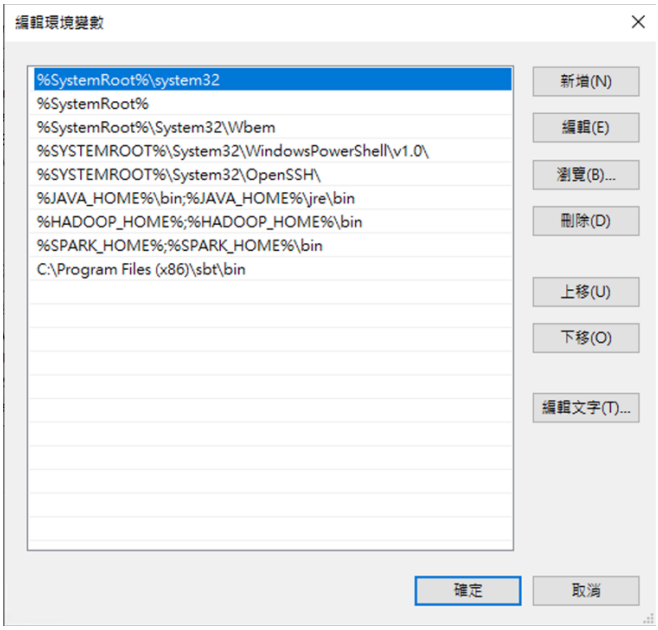


在 PATH 中加入

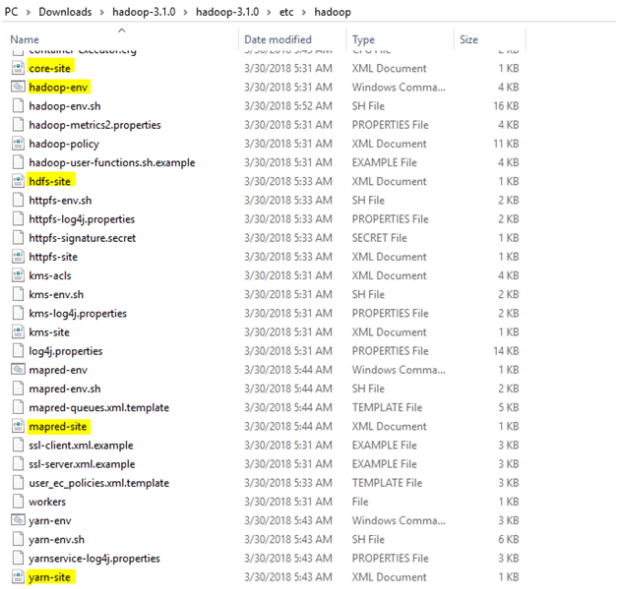
%JAVA_HOME%\bin;%JAVA_HOME%\jre\bin

%HADOOP_HOME%;%HADOOP_HOME%\bin

%SPARK_HOME%;%SPARK_HOME%\bin



配置 hadoop 文件（編輯 C:\hadoop-2.9.2\etc\hadoop 下的四個檔案）：



1. 編輯 C:\hadoop-2.9.2\etc\hadoop 下的 core-site.xml

```
<configuration>
    <property>
        <name>hadoop.tmp.dir</name>
        <value>/C:/hadoop/workplace/tmp</value>
    </property>
    <property>
        <name>dfs.name.dir</name>
        <value>/C:/hadoop/workplace/name</value>
    </property>
    <property>
        <name>fs.default.name</name>
        <value>hdfs://localhost:9000</value>
    </property>
</configuration>
```

2. 編輯 C:\hadoop-2.9.2\etc\hadoop 下的 mapred-site.xml

```xml
<configuration>
    <property>
        <name>mapreduce.framework.name</name>
        <value>yarn</value>
    </property>
    <property>
        <name>mapred.job.tracker</name>
        <value>hdfs://localhost:9001</value>
    </property>
</configuration>
```

3. 編輯 C:\hadoop-2.9.2\etc\hadoop 下的 hdfs-site.xml

```xml
<configuration>
    <property>
        <name>dfs.replication</name>
        <value>1</value>
    </property>
    <property>
        <name>dfs.data.dir</name>
        <value>/d:/hadoop/workplace/data</value>
    </property>
</configuration>
```

4. 編輯 C:\hadoop-2.9.2\etc\hadoop 下的 yarn-site.xml

```xml
<configuration>
<!-- Site specific YARN configuration properties -->
    <property>
        <name>yarn.nodemanager.aux-services</name>
        <value>mapreduce_shuffle</value>
    </property>
    <property>
        <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
        <value>org.apache.hadoop.mapred.ShuffleHandler</value>
    </property>
    <property>
        <name>yarn.resourcemanager.address</name>
        <value>140.113.72.196:8032</value>
    </property>
    <property>
        <name>yarn.resourcemanager.scheduler.address</name>
        <value>140.113.72.196:8030</value>
    </property>
    <property>
        <name>yarn.resourcemanager.resource-tracker.address</name>
        <value>140.113.72.196:8035</value>
    </property>
    <property>
        <name>yarn.resourcemanager.admin.address</name>
        <value>140.113.72.196:8033</value>
    </property>
    <property>
        <name>yarn.resourcemanager.webapp.address</name>
        <value>140.113.72.196:8088</value>
    </property>
</configuration>
```

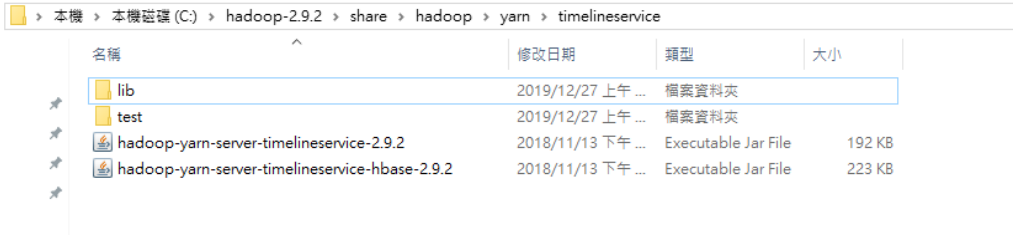
C:\hadoop-2.9.2\sbin 裡的 start-all.cmd 前面加 cd path 導至 bin

```
20
21    echo This script is Deprecated. Instead use start-dfs.cmd and start-yarn.cmd
22    cd C:\hadoop-2.9.2\bin
```

若沒有加上導至 bin 的程式碼，再後面的步驟就會一直出現以下截圖的錯誤

```
C:\hadoop-2.9.2\sbin>start-all.cmd
This script is Deprecated. Instead use start-dfs.cmd and start-yarn.cmd
系統找不到檔案 hadoop。
系統找不到檔案 hadoop。
starting yarn daemons
系統找不到檔案 yarn。
系統找不到檔案 yarn。
```

複製 C:\hadoop-2.9.2\share\hadoop\yarn\timelineservice\hadoop-yarn-server-timelineservice-2.9.2 到上一層 C:\hadoop-2.9.2\share\hadoop\yarn

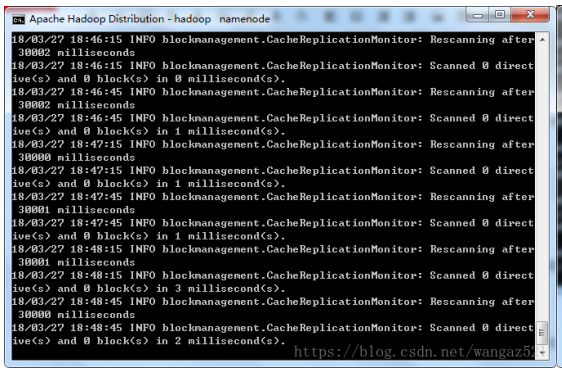


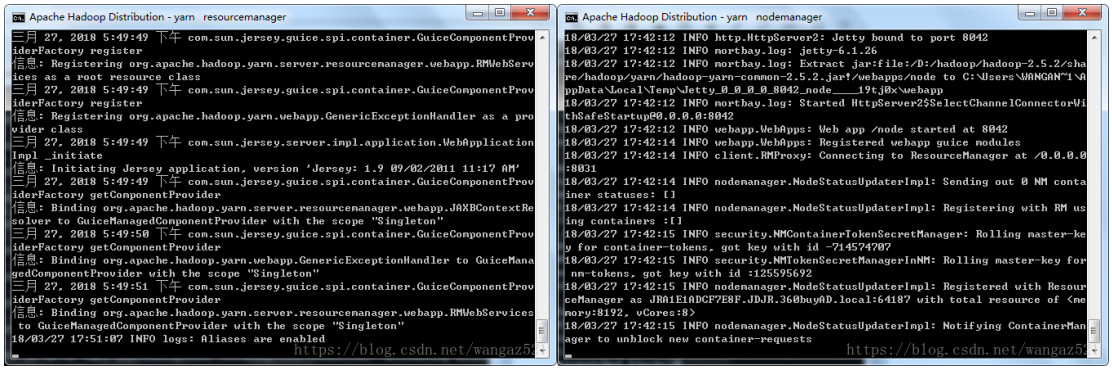
若沒有上面步驟，會出現 resource manager 在執行一小段時間後就關掉的狀況。這裡透過 export hadoop yarn log 才觀察到的問題。

以上設定完成後先至 bin 底下下 hdfs namenode –format 指令

```
C:\hadoop-2.9.2\bin>hdfs namenode -format
19/12/27 10:24:48 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = DESKTOP-I6NJKCD/140.113.72.196
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 2.9.2
STARTUP_MSG:   classpath = C:\hadoop-2.9.2\etc\hadoop;C:\hadoop-2
op-2.9.2\share\hadoop\common\lib\apacheds-i18n-2.0.0-M15.jar;C:\
```

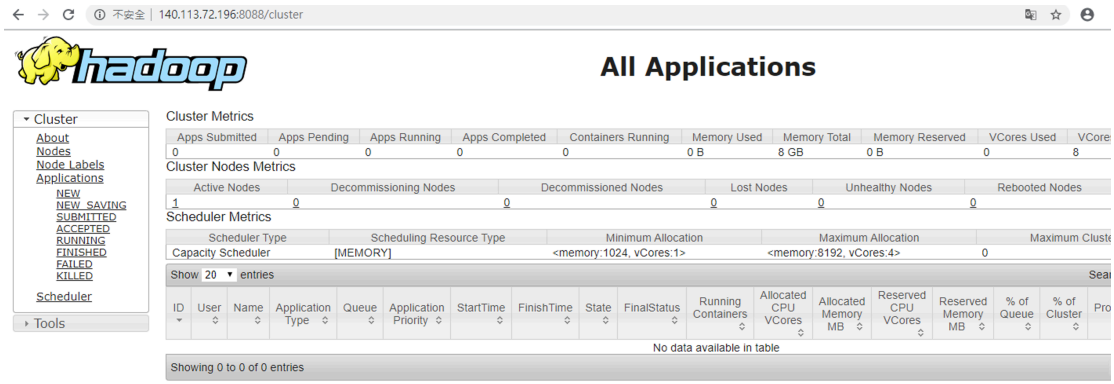
在切換到 sbin 目錄執行 start-all.cmd，會啟動四個 process，並跳出以下四個視窗：

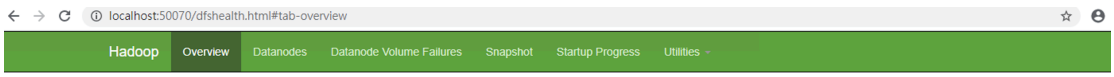




在 cmd 到 java\jdk-version\bin 底下下 jps 確認正在執行的 process

開啟網頁確認 localhost:8088 跟 localhost:50070 有出現以下兩個畫面





Overview 'localhost:9000' (active)

Started:	Fri Dec 27 11:51:09 +0800 2019
Version:	2.9.2, r826afbeae31ca687bc2f8471dc841b66ed2c6704
Compiled:	Tue Nov 13 20:42:00 +0800 2018 by ajisaka from branch-2.9.2
Cluster ID:	CID-ffa63578-c4cb-43d1-895a-d571e593728f
Block Pool ID:	BP-968912654-140.113.72.196-1577418654922

Summary

Security is off.

Safemode is off.

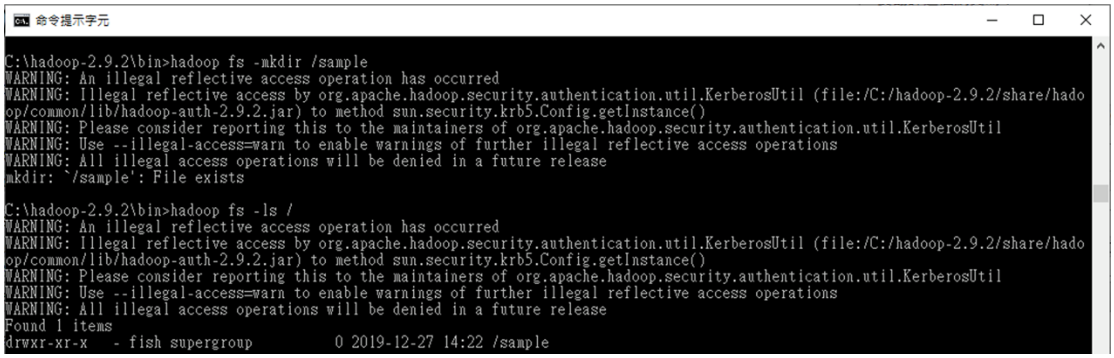
1 files and directories, 0 blocks = 1 total filesystem object(s).

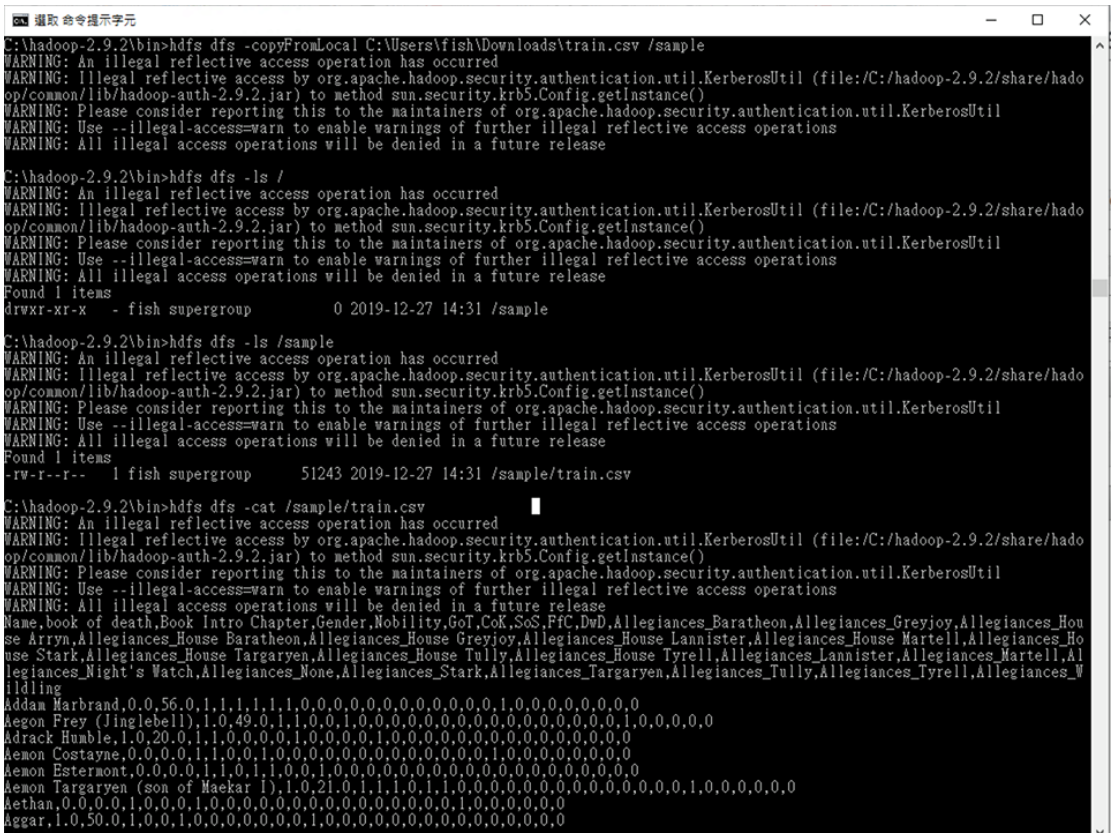
Heap Memory used 76.4 MB of 117 MB Heap Memory. Max Heap Memory is 1000 MB.

Non Heap Memory used 36.73 MB of 40.38 MB Commited Non Heap Memory. Max Non Heap Memory is <unbounded>.

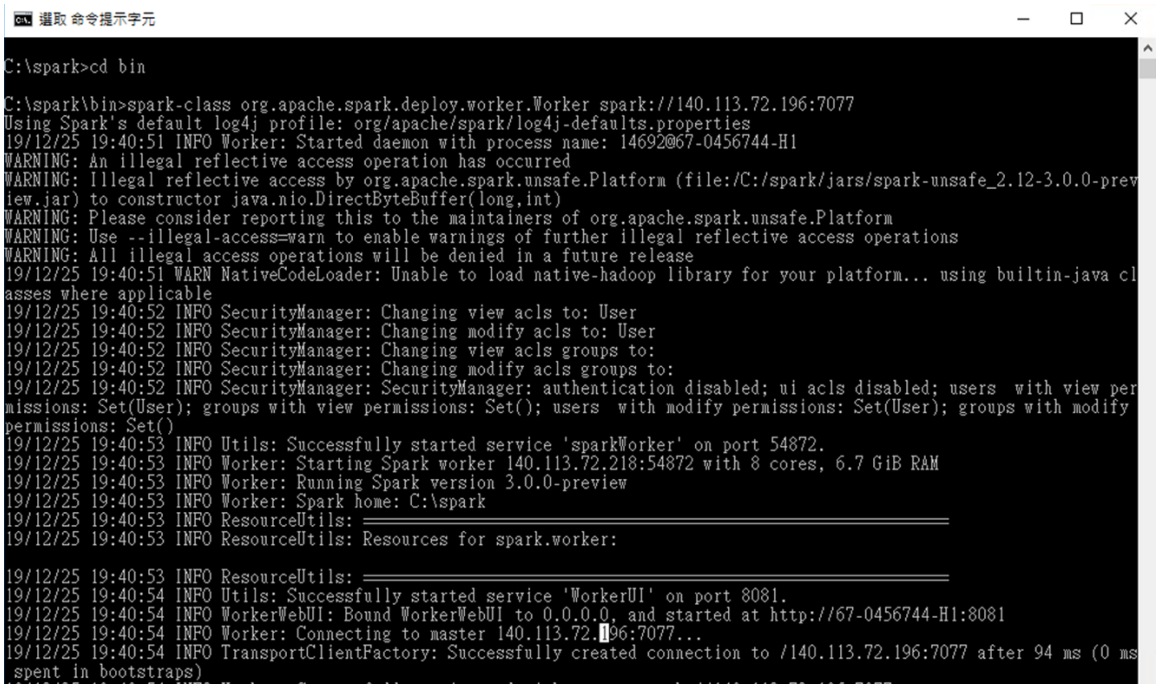
Configured Capacity:	0 B
DFS Used:	0 B (100%)
Non DFS Used:	0 B

連接 hdfs，創建目錄並丟檔案到上面，並測試檔案有辦法下載

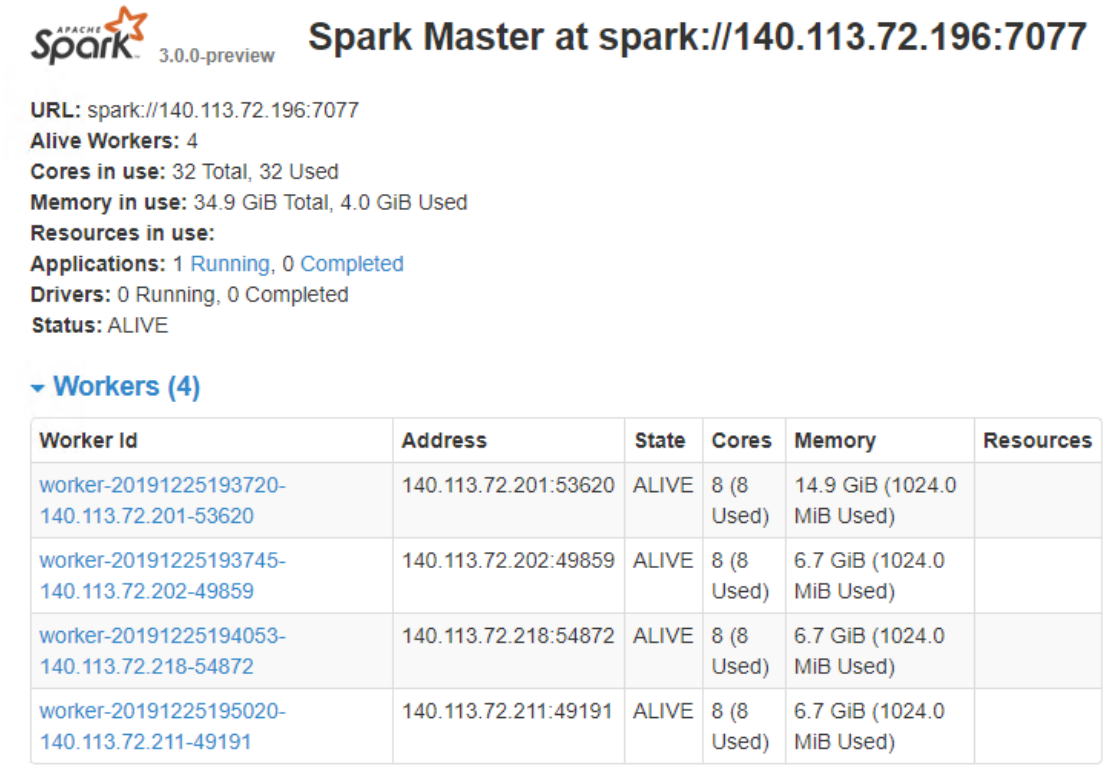




（三）spark 連接 worker 和 master(140.113.72.196)



（四）將 worker 連接 master(140.113.72.196)後完成連接的畫面(localhost:8080)



二、作業目標對應之結果

（一）將資料讀取進來(可用 pandas 套件)、資料前處理 、把空值以 0 替代

```
# read files
global Path
sc = SparkContext()
if sc.master[0:5]=='local': #代表由本機執行，直接讀取本機檔案
    Path="file:///"
else: #在cluster執行，讀取hdfs檔案
    Path="hdfs://master:9000/sample/"

df = pd.read_csv(Path+'character-deaths.csv')

# fill blanks
df = df.fillna(0)
```

（二） Death Year , Book of Death , Death Chapter 三者取一個，將有數值的轉成 1、將 Allegiances 轉成 dummy 特徵(底下有幾種分類就會變成幾個特徵，值是 0 或 1，本來的資料集就會再增加約 20 種特徵)

```
# turn num into 1
bookofdeath = np.zeros(df.shape[0])
for i in range(df.shape[0]):
    if (df['Book of Death'][i] != 0):
        bookofdeath[i] = 1
df.insert(2,'book of death',bookofdeath)

# drop Death Year, Death Chapter
df = df.drop("Book of Death", axis=1)
df = df.drop("Death Year", axis=1)
df = df.drop("Death Chapter", axis=1)

# 將Allegiances轉成dummy
dummies_1 = pd.get_dummies(df, columns=['Allegiances'], drop_first=True)
df = dummies_1
```

（三）亂數拆成訓練集(75%)與測試集(25%) 把處理好的資料存進 train.csv、test.csv

```
# feature target
target = df.loc[:, ['book of death']]
df_feature = df.drop("book of death", axis=1)
features = df_feature

# 亂數拆成訓練集(75%)與測試集(25%)
#x_train, x_test, y_train, y_test = train_test_split(features, target, test_size=0.25)
#train=pd.concat([x_train,y_train],axis=0)
msk = np.random.rand(len(df)) < 0.75
train = df[msk]
test = df[~msk]

#sp_train = sparkSession.createDataFrame(train)
#sp_train.write_csv('hdfs://cluster/user/hdfs/train.csv')
#sp_test = sparkSession.createDataFrame(test)
#sp_test.write_csv('hdfs://cluster/user/hdfs/train.csv')

train.to_csv('train.csv',index=None,header=True)
test.to_csv('test.csv',index=None,header=True)
```

（四）讀資料並把資料轉成 RDD 檔

```
train_rawDataWithHeader=sc.textFile(Path+'train.csv')
test_rawDataWithHeader=sc.textFile(Path+'test.csv')
#train_data=prepare_data(train)
header = train_rawDataWithHeader.first()
rawData = train_rawDataWithHeader.filter(lambda x: x != header)
rData = rawData.map(lambda x: x.replace("\"", ""))
train_data = rData.map(lambda x: x.split(","))
#test_data=prepare_data(test)
header = test_rawDataWithHeader.first()
rawData = test_rawDataWithHeader.filter(lambda x: x != header)
rData = rawData.map(lambda x: x.replace("\"", ""))
test_data = rData.map(lambda x: x.split(","))

lines=train_data+test_data
categoriesMap = lines.map(lambda fields: fields[0]).distinct().zipWithIndex().collectAsMap()
train_RDD=train_data.map(lambda r:LabeledPoint(extract_label(r),extract_features(r,categoriesMap,len(r)-1)))
test_RDD=test_data.map(lambda r: (extract_features(r,categoriesMap,len(r)-1),r[-1]))
```

（五）建立決策樹模型、產出預測結果、計算 accuracy, recall, precision

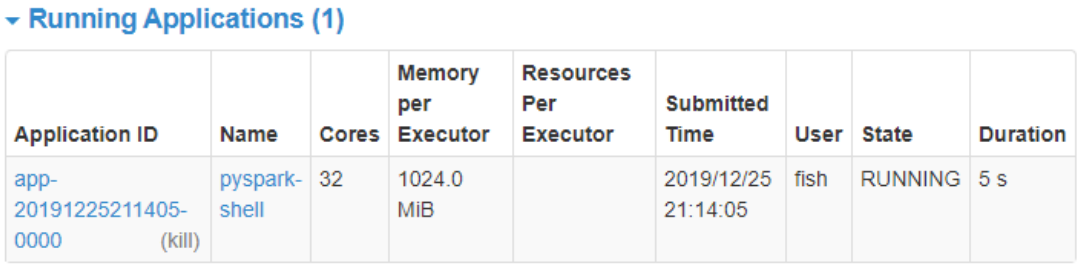
```
model=DecisionTree.trainClassifier(train_RDD,numClasses=2,categoricalFeaturesInfo={},impurity='entropy',maxDepth=14,maxBins=9)

count = 0
num = 0
positive = 0
negative = 0
truePositive = 0
trueNegative =0
falsePositive = 0
falseNegative = 0
for data in test_RDD.take(test_data.count()):
    num+=1
    preds = int(model.predict(data[0]))
    #print(str(preds)+' '+str(data[1]))
    if(preds == int(data[1])):count=count+1
    if int(data[1]) == 0: negative += 1
    if int(data[1]) == 1: positive += 1
    if (int(preds) == 1 and int(data[1]) == 1) : truePositive +=1
    if (int(preds) == 0 and int(data[1]) == 1) : falseNegative +=1
    if (int(preds) == 0 and int(data[1]) == 0) : trueNegative +=1
    if (int(preds) == 1 and int(data[1]) == 0) : falsePositive +=1
acc = count/num
precision = float(truePositive) / (float(truePositive) + float(falsePositive))
recall =  float(truePositive) / (float(truePositive) + float(falseNegative))
print('Accuracy:'+str(acc))
print('Precision:'+str(precision))
print('Recall:'+str(recall))
```

（六）執行程式

需要先導入 pyspark

使用 pycharm 執行以上程式

執行期間打開 localhost:8080 可以看到目前有 Spark shell 的應用正在運行（如下圖所示）：



（七）實際產出結果

```
Accuracy:0.9591836734693877
Precision:0.5
Recall:0.8
```

討論：

我們直接將 spark 裝在 windows 系統上，因此跟書上步驟不同，過程有遇到不同的問題，不過最後還是可以順利預測結果。

參考資料：

https://kknews.cc/zh-tw/code/nrxbg52.html
http://www.lawrencelin.info/2017/11/spark-on-yarn-in-windows/
https://www.cnblogs.com/chevin/p/9090683.html
https://zhuanlan.zhihu.com/p/29362852
https://blog.csdn.net/wangaz521/article/details/79717177
https://www.codementor.io/@jadianes/spark-python-mllib-decision-trees-du107qr0j