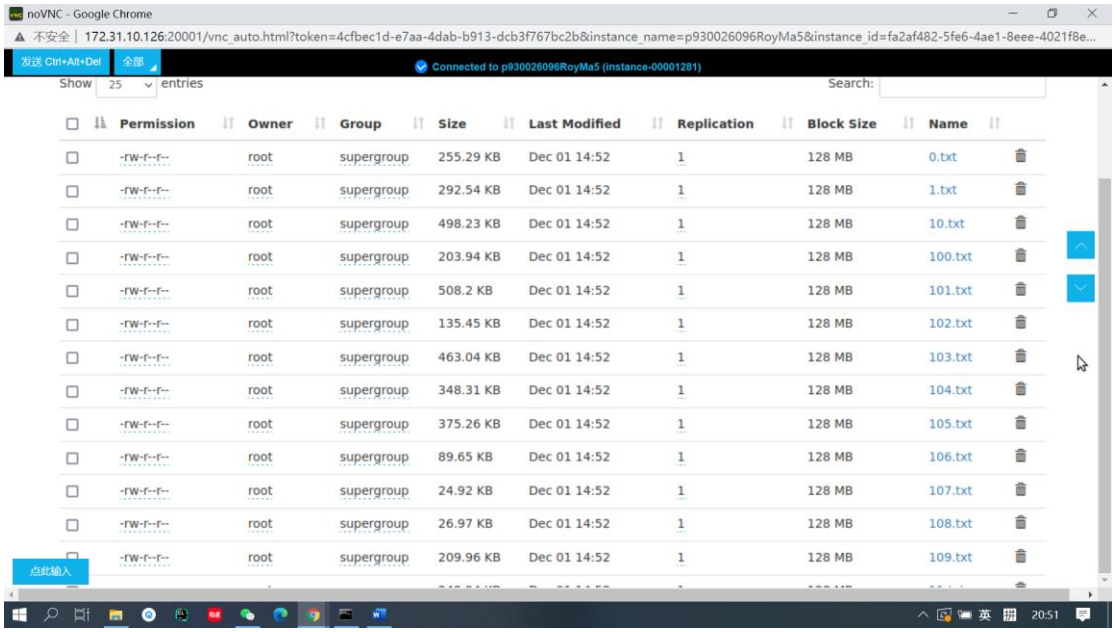
Intermediate

```
In [1]: from pyspark import SparkConf, SparkContext
        from pyspark.sql import SparkSession, SQLContext
        import os, re
        import math

        #First we compute the TF-IDF of all files in ./iitfidf, save as an RDD
        #We need SQLContext for processing RDD<->DF
        #If you want to run the job on Spark cluster, you have to modify the following line, e.g.:
        #spark = SparkSession.builder.master('spark://dpw2tcxu:7077').appName('MyWordCount').getOrCreate()
        #sc = spark.sparkContext
        sc = SparkContext.getOrCreate(SparkConf())
        sql = SQLContext(sc)

        #If you want to run the job with Hadoop HDFS, you have to modify the following line, e.g.:
        data = sc.wholeTextFiles('hdfs://10.20.4.50:9000/user/hduser/input/books_306_121M')
        #data = sc.wholeTextFiles('hdfs://10.20.4.50:9000/user/hduser/input/books_2528_1000M/*.txt')
        #data = sc.wholeTextFiles('./iitfidf')

        numFiles = data.count()

        wordcount = data.flatMap(lambda x: [((os.path.basename(x[0]) ,i) ,1) for i in re.split('\\W', x[1])])\
                .reduceByKey(lambda a, b: a + b)
        wordcount.collect()

        tf = wordcount.map(lambda x: (x[0][1],(x[0][0],x[1])))

        idf = wordcount.map(lambda x: (x[0][1], (x[0][0], x[1], 1)))\
                .map(lambda x: (x[0], x[1][2]))\
                .reduceByKey(lambda x, y: x + y)\
                .map(lambda x: (x[0], math.log10(numFiles / x[1])))

        #Slightly modified map output as (doc, (term, tfidf))
        tfidf = tf.join(idf)\
                    .map(lambda x: (x[1][0][0], (x[0], x[1][0][1] * x[1][1])))\
                    .sortByKey()
```

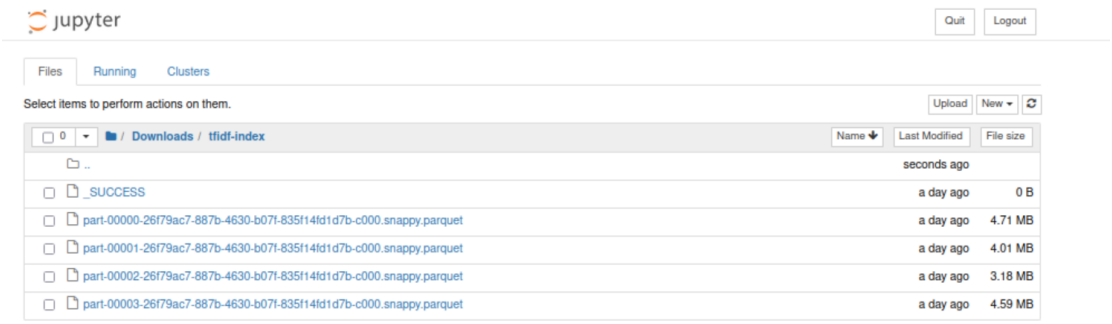
Change the directory to own HDFS server and based on the documents in HDFS.



Here is the files in our own HDFS

```
#Slightly modified map output as (doc, (term, tfidf))
tfidf = tf.join(idf)\
            .map(lambda x: (x[1][0][0], (x[0], x[1][0][1] * x[1][1])))\
            .sortByKey()

#Then we convert the TF-IDF to an DF, and save to the disk
lines = tfidf.map(lambda x: (x[0], x[1][0], x[1][1])).toDF()
lines.write.save("tfidf-index")
```

Calculate the tf-idf and save the files in a file directory



Here is the files that generated by the codes