***DS 3003***

**Data Processing Workshop II**

**Group Project Report**

**Group 13**

**Mengxian HE (Kris) 1830026031**

**Yuanhao LI (Erwin) 1830026070**

**Jianlong LI (Draco) 1830026061**

## Data collection and clean

First, we begin to collect data, we write a spider “data\_collect.py” to scrip data from <http://www.gutenberg.org/ebooks/search/?sort_order=release_date> and clean data.

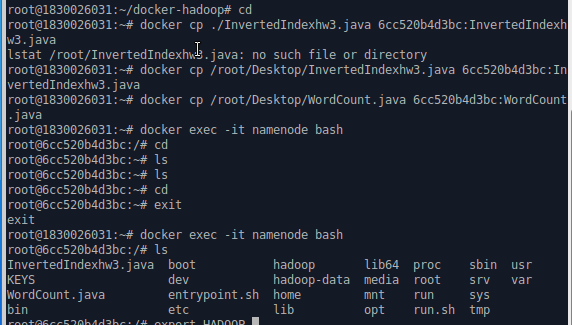
The spider has six functions, download,update,merge,clean, toxml and x2j. Download function will download English e-books by size to “downPATH” folder, and write the largest and smallest id to "conf” file. Update function will get the latest e-book in local host by reading “conf” file and get the latest e-book in website, if there are different, it will download all new books. Merge function will merge all text file in a folder to one text file. Clean function will extract all English words in a file and add a id at the beginning. Toxml function will add a <text><id> tags to plain text file to convert it to a xml file. X2j functions can convert all xml file in a folder to json file. For details of it, please cheek the help by commend “python data\_collect.py -h” and the part ”1.data\_collect.py” of “README.md” in my github repository <https://github.com/walkureHHH/juner_1_dp2_porject1> .

## Inverted index by python

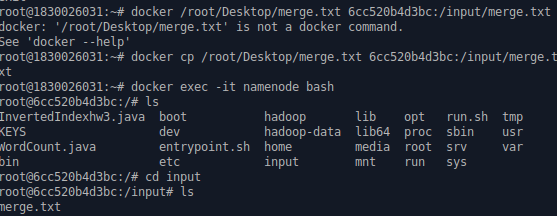
Than we will do inverted index by python, we should use a cleaned and merged text file whose split character between id and content is TAB. To do inverted index with python, I write two python file “mapper.py” and “reducer.py”. The usage and example please cheek the part “2.mapreduce\_python” of “README.md” in my github repository <https://github.com/walkureHHH/juner_1_dp2_porject1> .

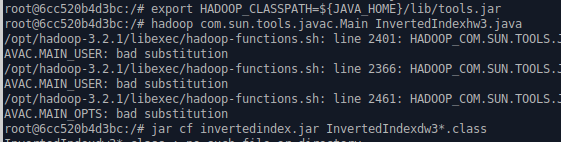
## Mapreduce in hadoop

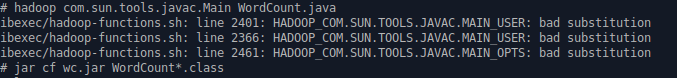
1.Check the container ID and copy the java file into the container.



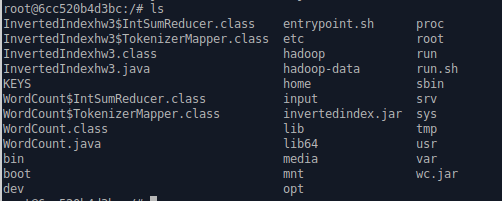
2.Copy the merge.txt into namenode

3. Compile the java file into class file

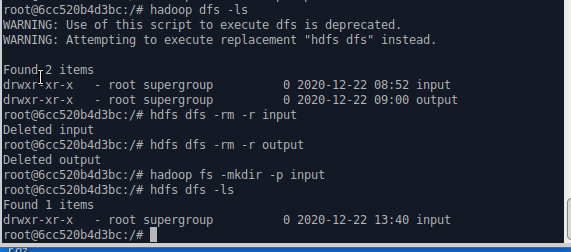




Succeed complied



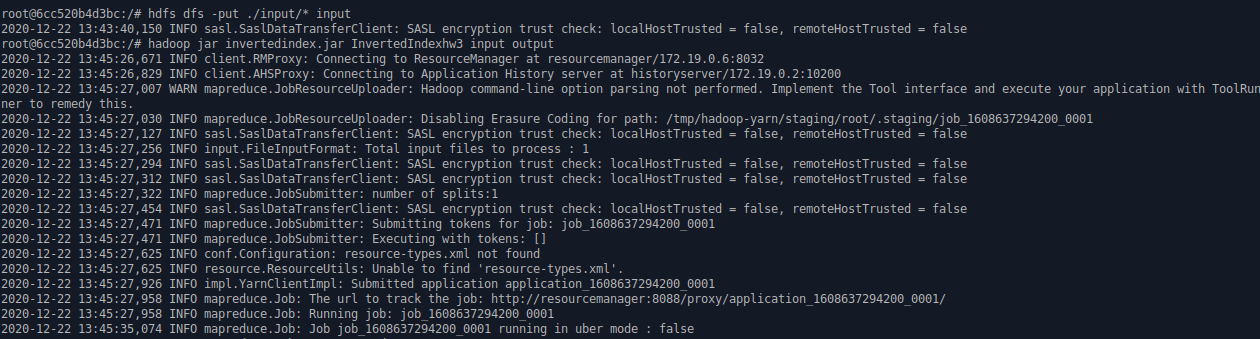
4.set the path



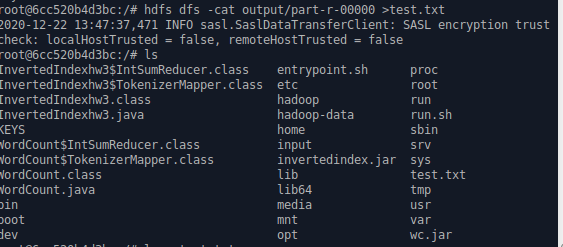


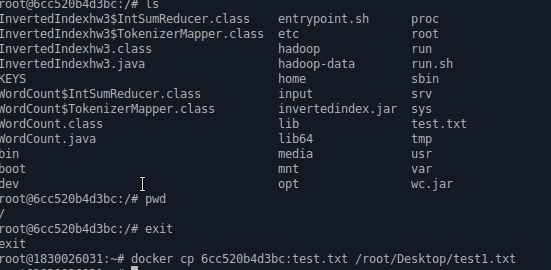
5.Execute

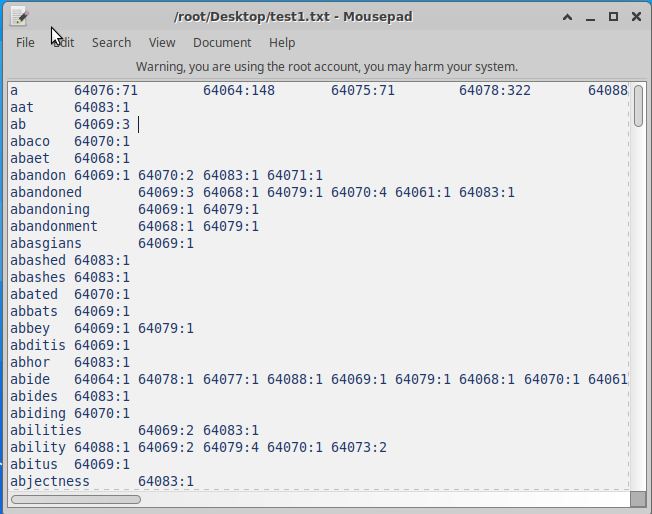
Input:



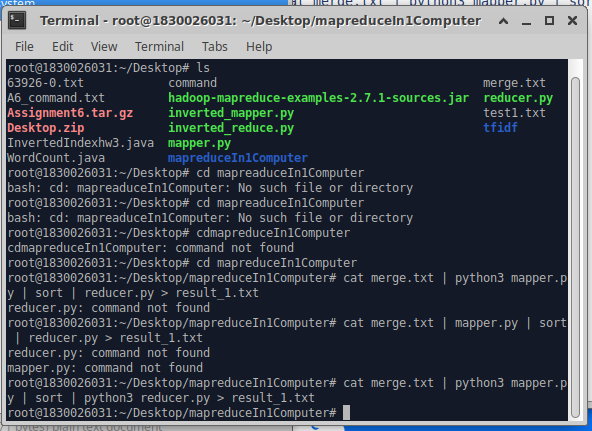
Output: as a test.txt



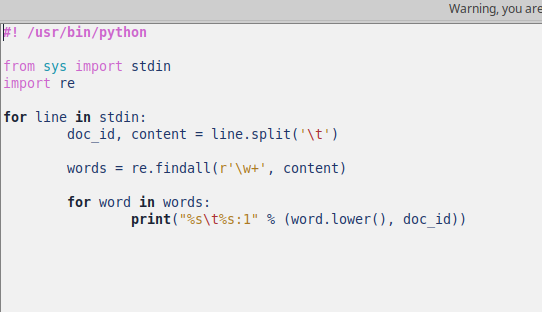




MapReduce by Python



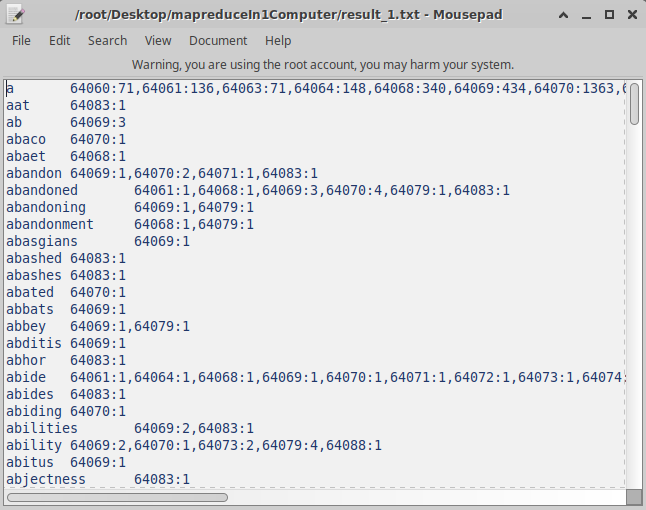
Mapper.py - source code



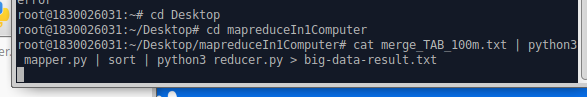
Reduce.py source code



The result was stored as txt:

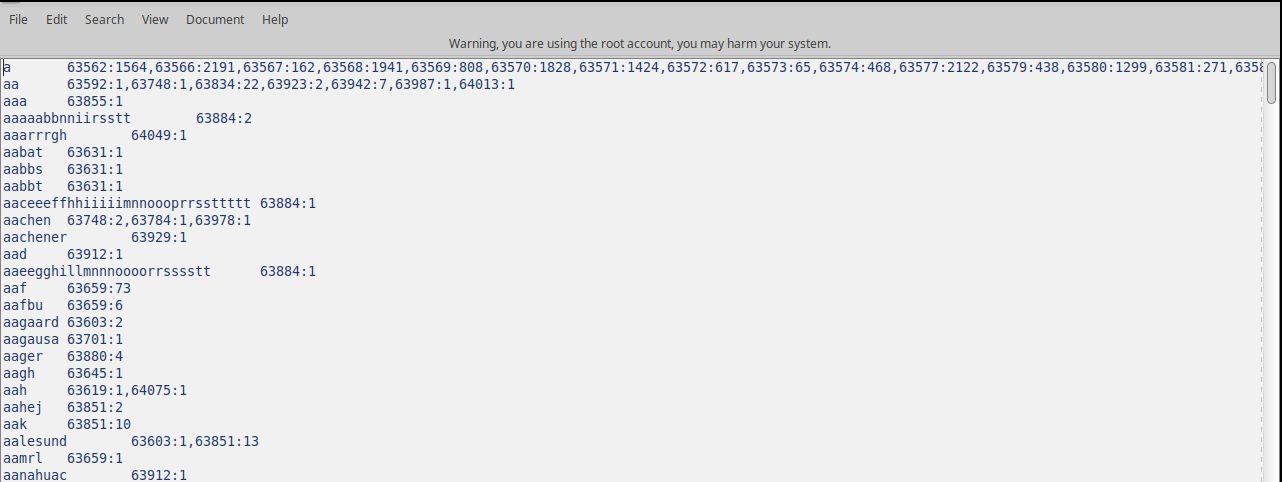


Deal with big-data (over 100m)



little slow (33.2s)......

But still get the result

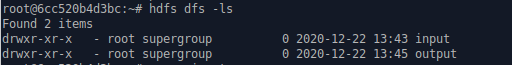


Mapreduce in hadoop:

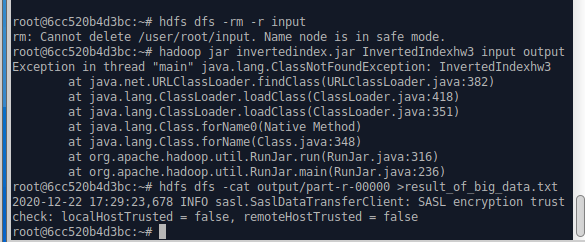
Copy the file in java:

1608658217(1)

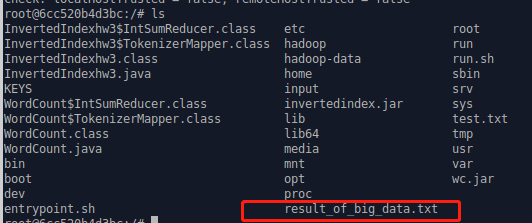
Check the connection path:



Deal with InvertedIndexhw3 with namenode and its three slave:

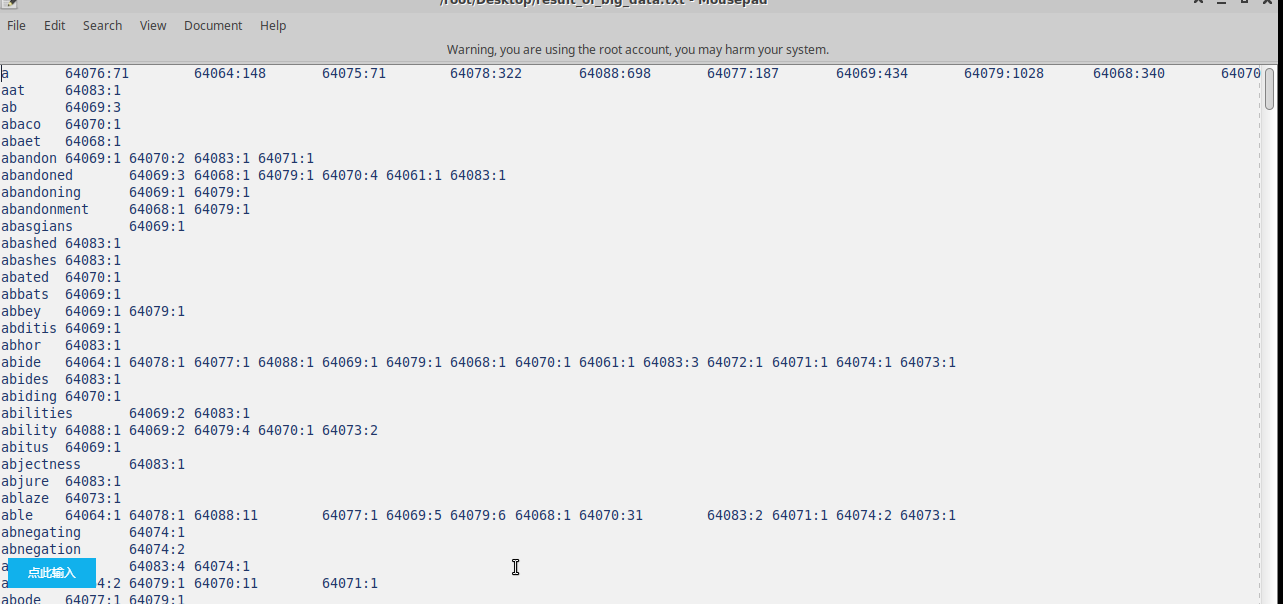


Output as a txt.file named result\_of\_big\_data.txt



Copy it from container to desktop and check the result:

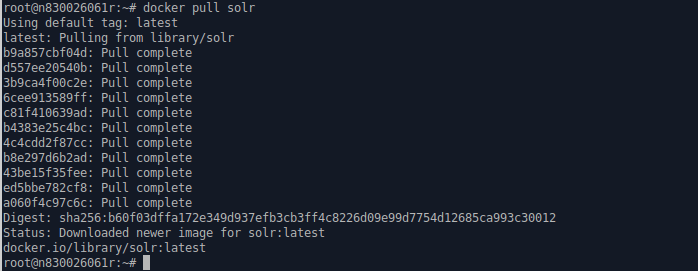
1608658664(1)



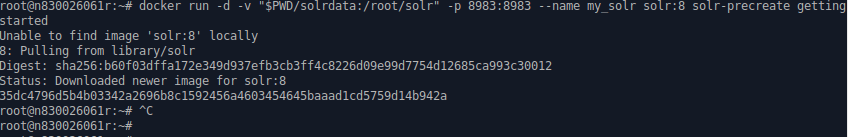
The speed of hadoop is much faster than the Mapreduce by python!

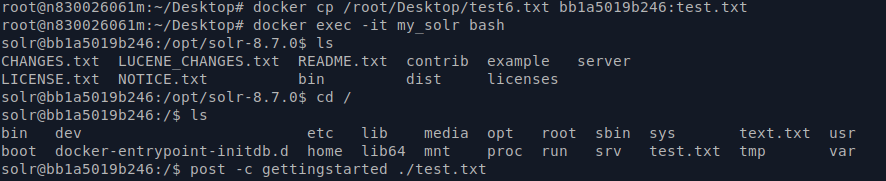
For improve the perfermance of map and reduce, we decide use solr and hbase to decrease the search time.

#### Solr part

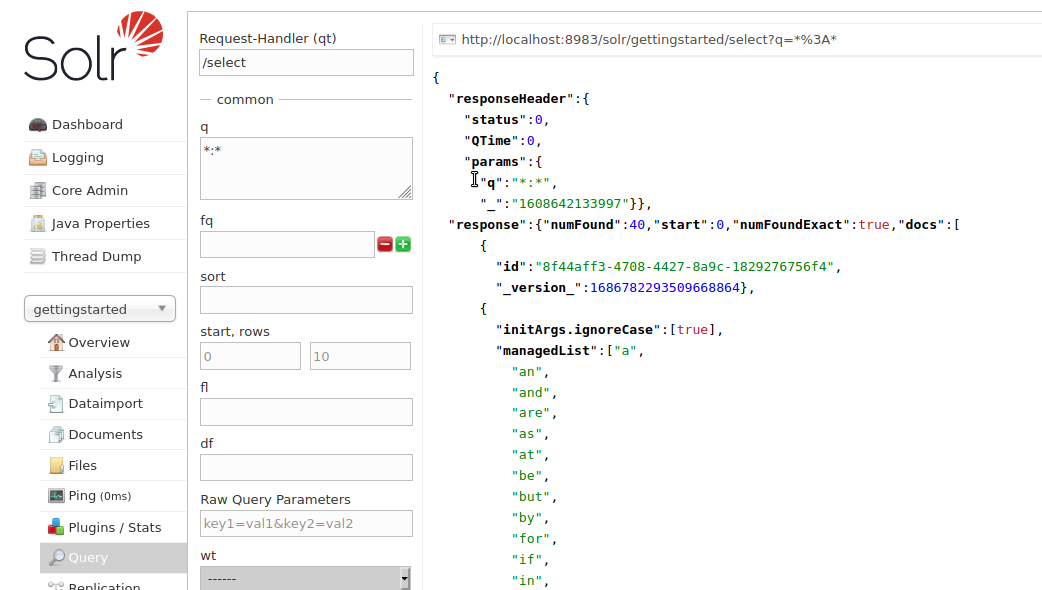


Firstly we pull solr from docker.





We put the text document into the Docker container and execute.



See the Solr interface through the Firefox in the TStack Tencent.

The interface of Solr and above we use the Query of select all operation and comes out the result sorted by alphabetical order.

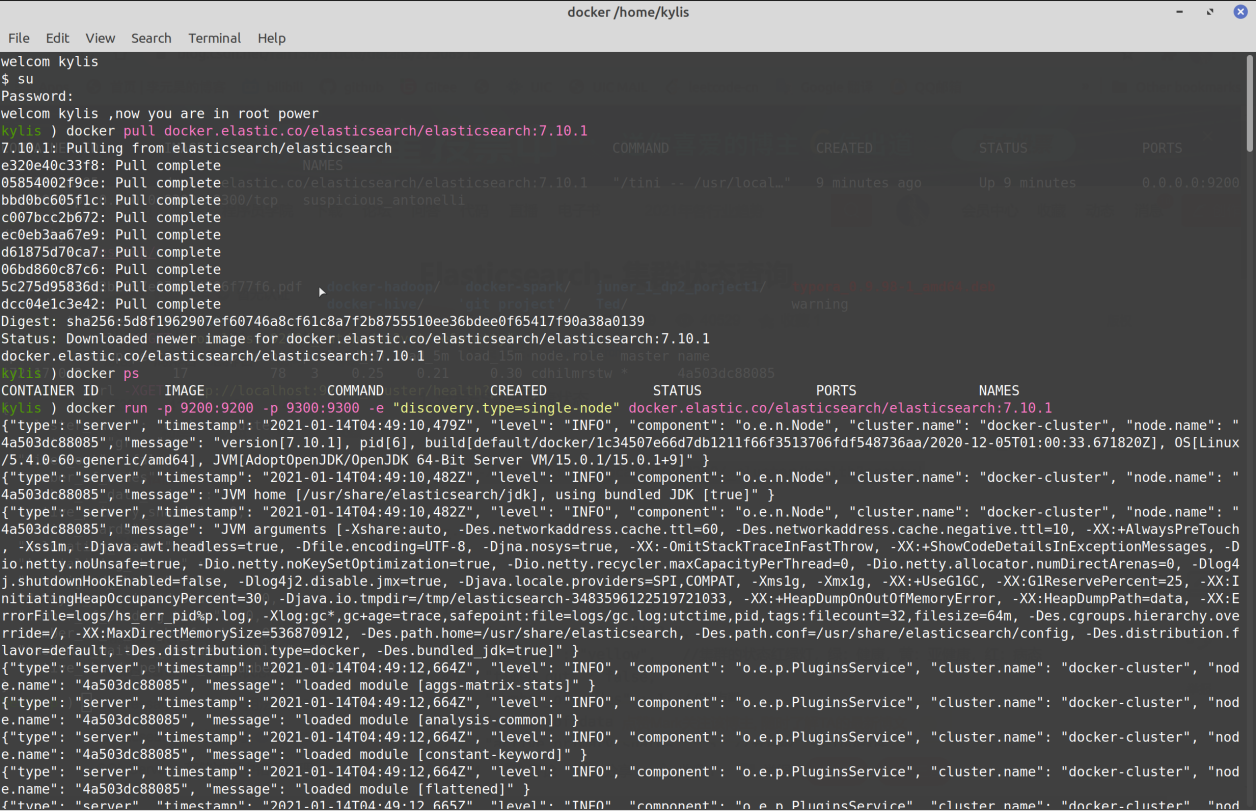
## Ranking by TFIDF

Than we will do search and ranking, I write the “search.py” to do this job. The file be searched should be cleaned and merged whose split character between id and content is SPACE. The usage and example please cheek the part “3.tfidf” of “README.md” in my github repository <https://github.com/walkureHHH/juner_1_dp2_porject1> .

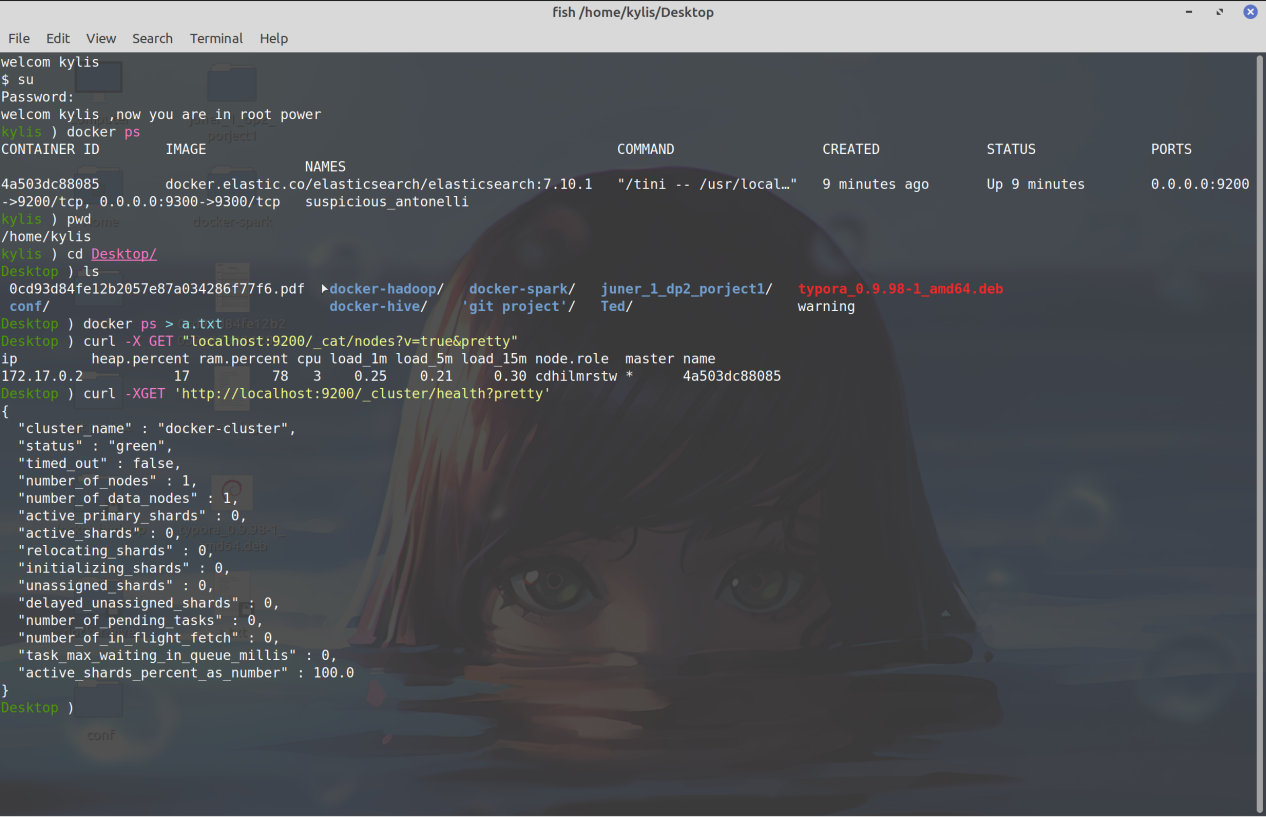
## Bonus

### elastic search

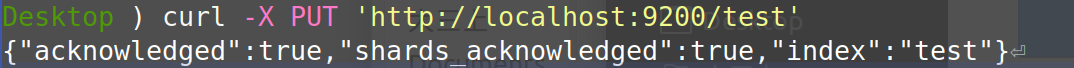
1. pull elasticsearch and run it



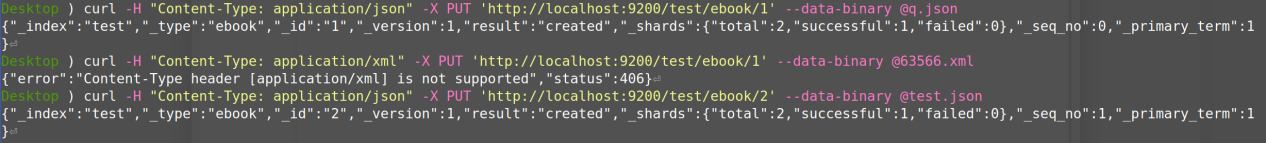
1. check the status



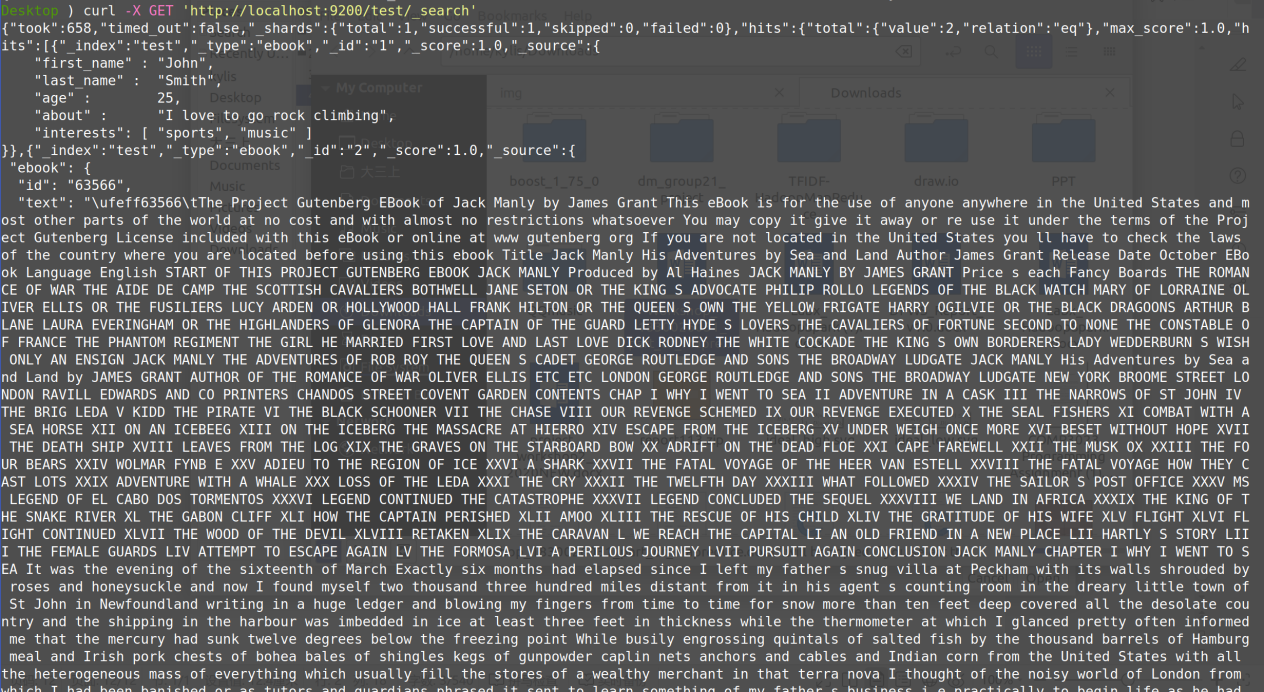
1. create a new index



1. import some json file



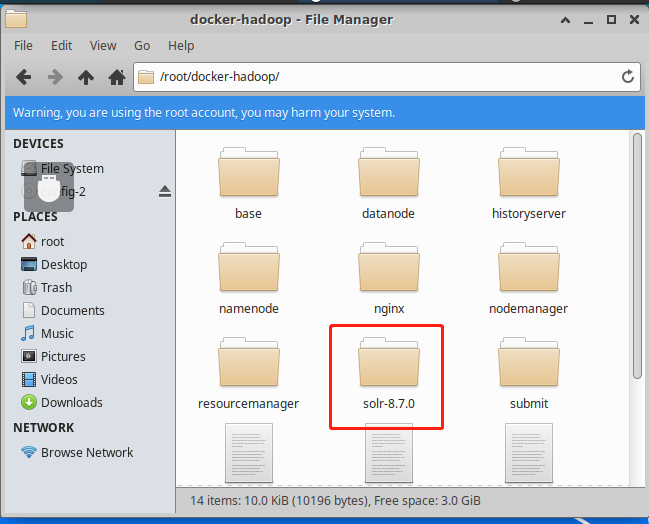
1. search the file we import



### Running solr in HDFS

1. install solr in Hadoop

New a folder called solr



Install solr in here and setting the path

1610637676(1)

1610637771(1)

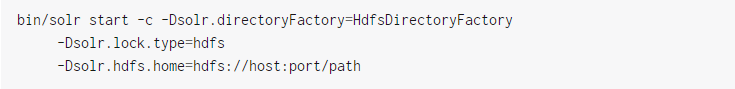
Save the modify

1610637875(1)

1. start the solr in following order:



1. sepecify the solr.hdfs.instance



1. the block cache

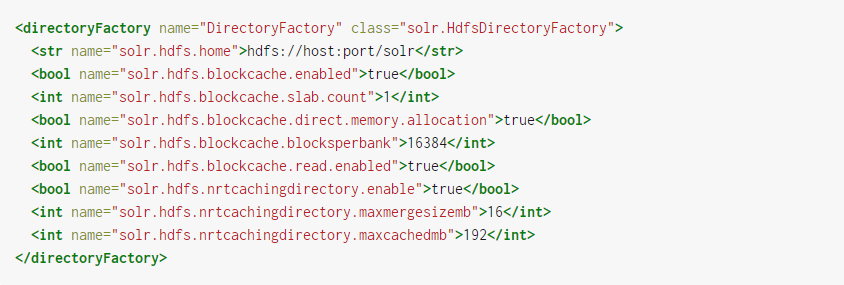
For performance, the HdfsDirectoryFactory uses a Directory that will cache HDFS blocks.

1610638051(1)

1. HdfsDirectoryFactory Parameters setting

Follow the document order...

1. Final config setting



## 7 . Contribution

Erwin: data collect and process, inverted index with python, search by ifidf, elastic search

Kris: mapreduce in hadoop

Draco: solr part, PPT