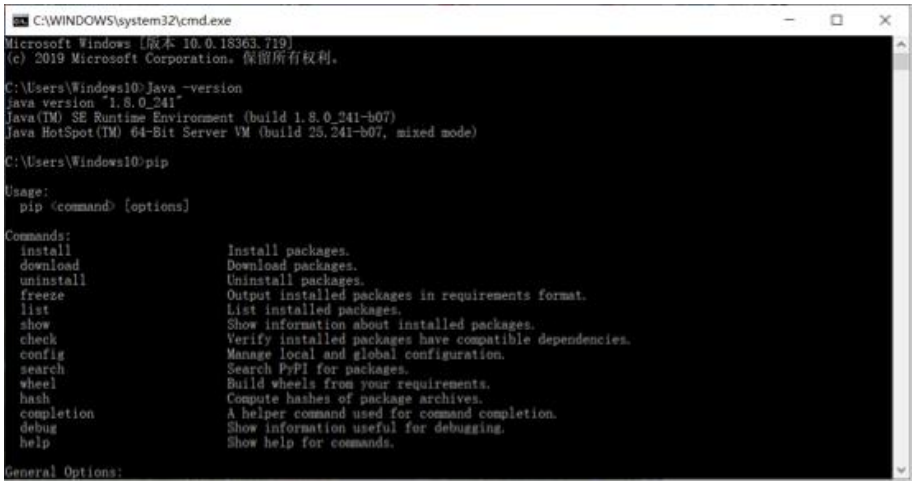
管理信息系统 NoSQL 实验报告

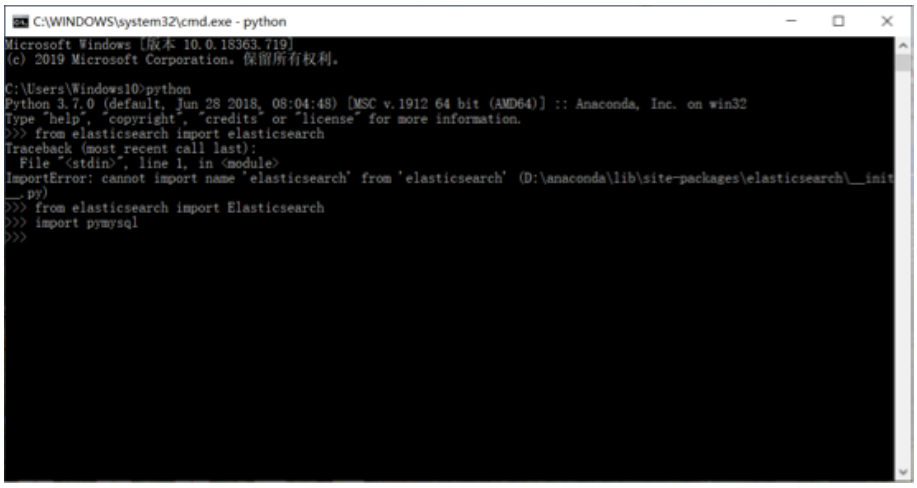
180812-17374463-王凯东

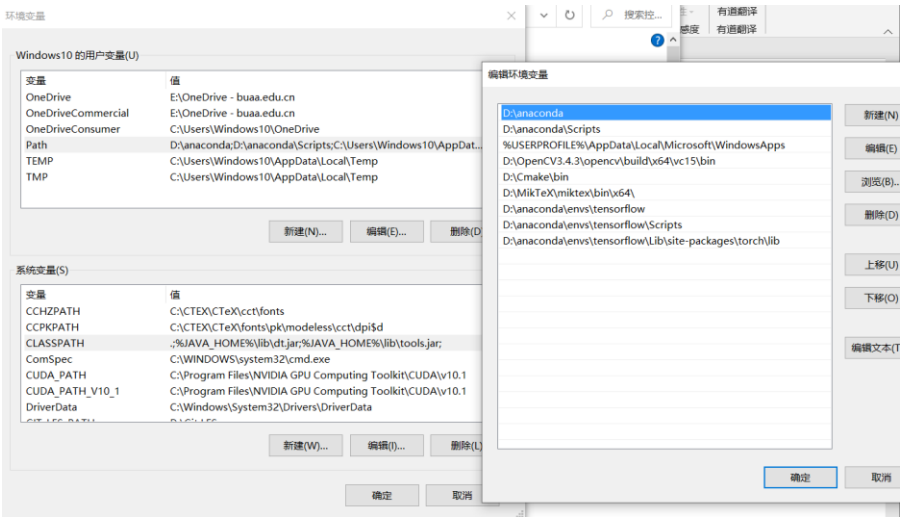
实验目的：

在本地计算机上使用 Python 完成示例数据集（news.txt）上 ES 和 MySQL 数据写入、查询过程，对比两种数据库的功能和性能差异，程序输出和效率。

1、安装 Java1.8、Python3.7.0 并成功配置环境变量



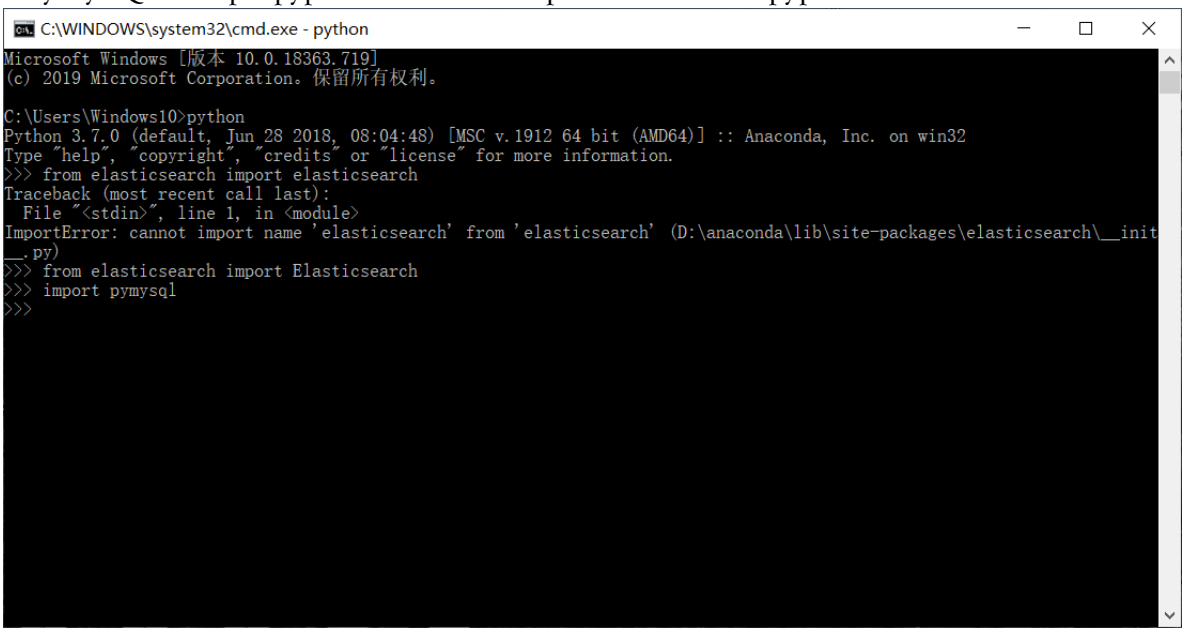




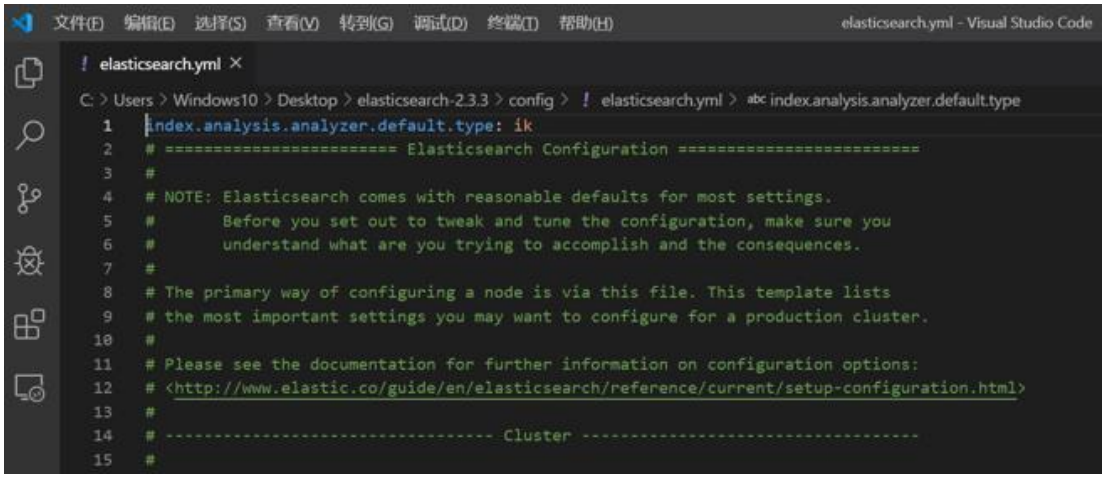
2、安装 elasticsearch 库和 pymysql 库

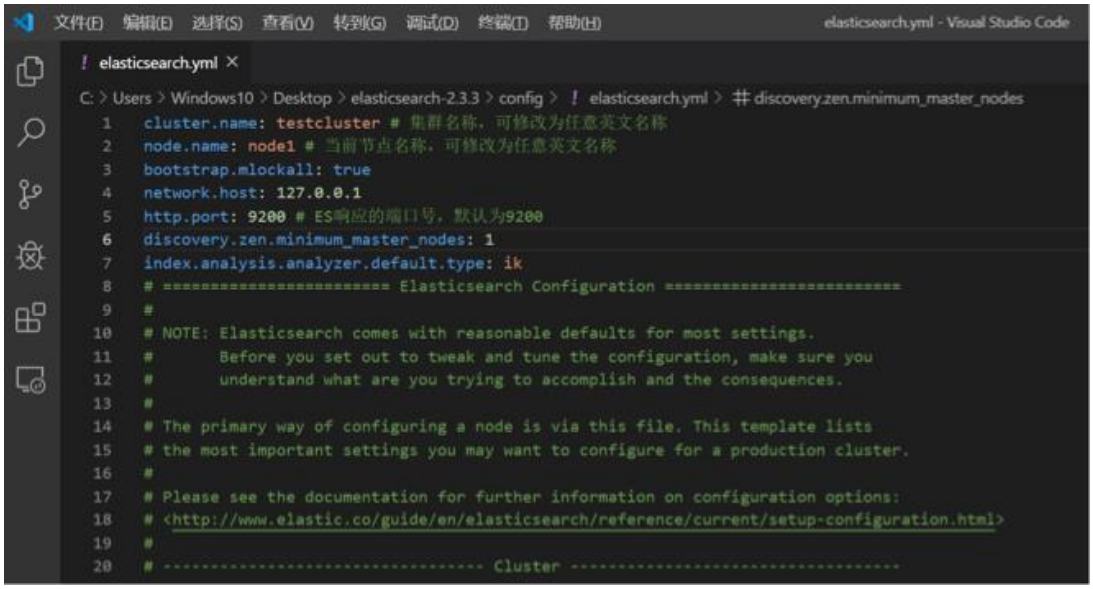
在命令行执行：

pip install elasticsearch==5.4.0 -i http://pypi.douban.com/simple --trusted-host pypi.douban.com;
pip install PyMySQL -i http://pypi.douban.com/simple --trusted-host pypi.douban.com

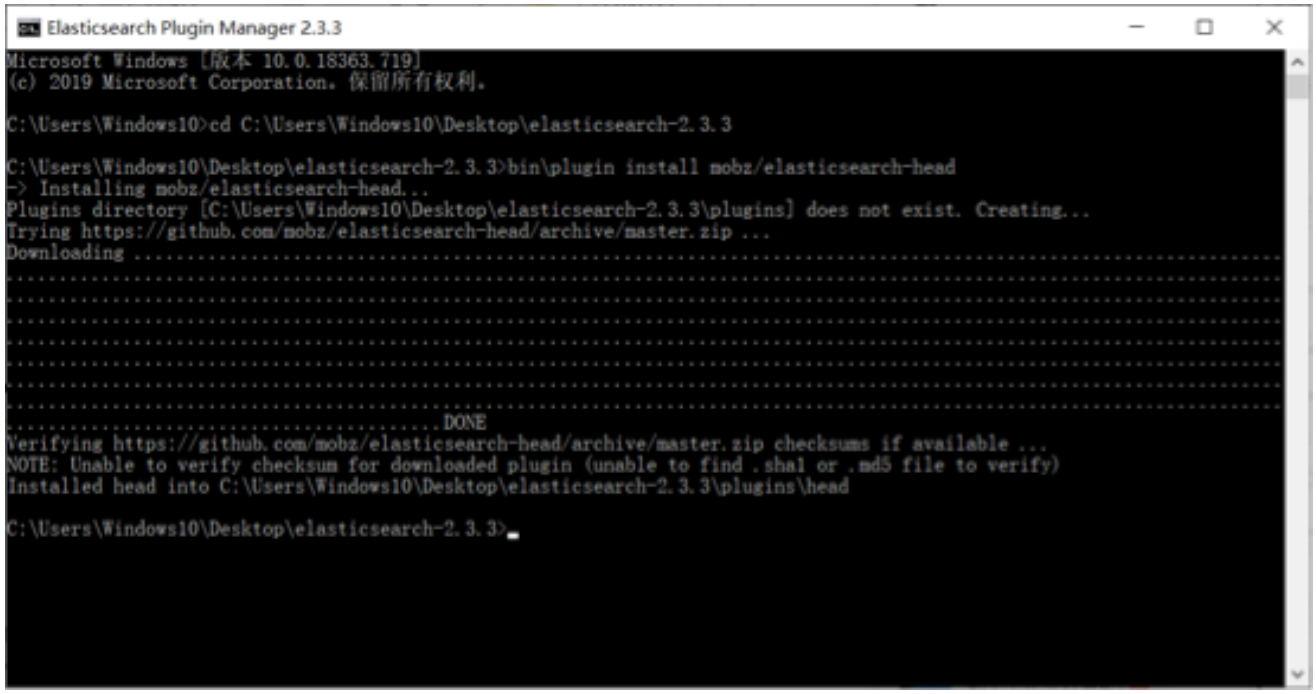


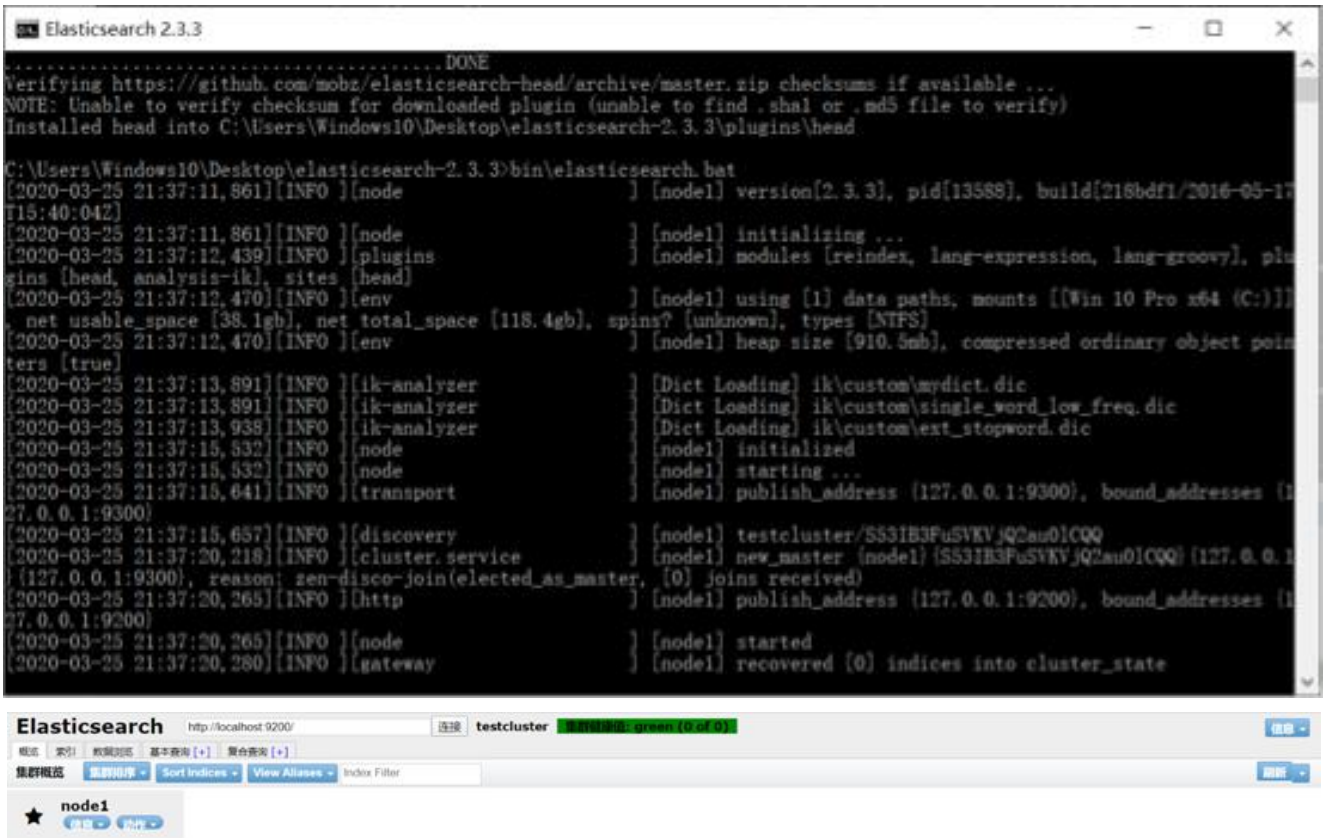
3、安装 IK 分词插件并创建结点 node1



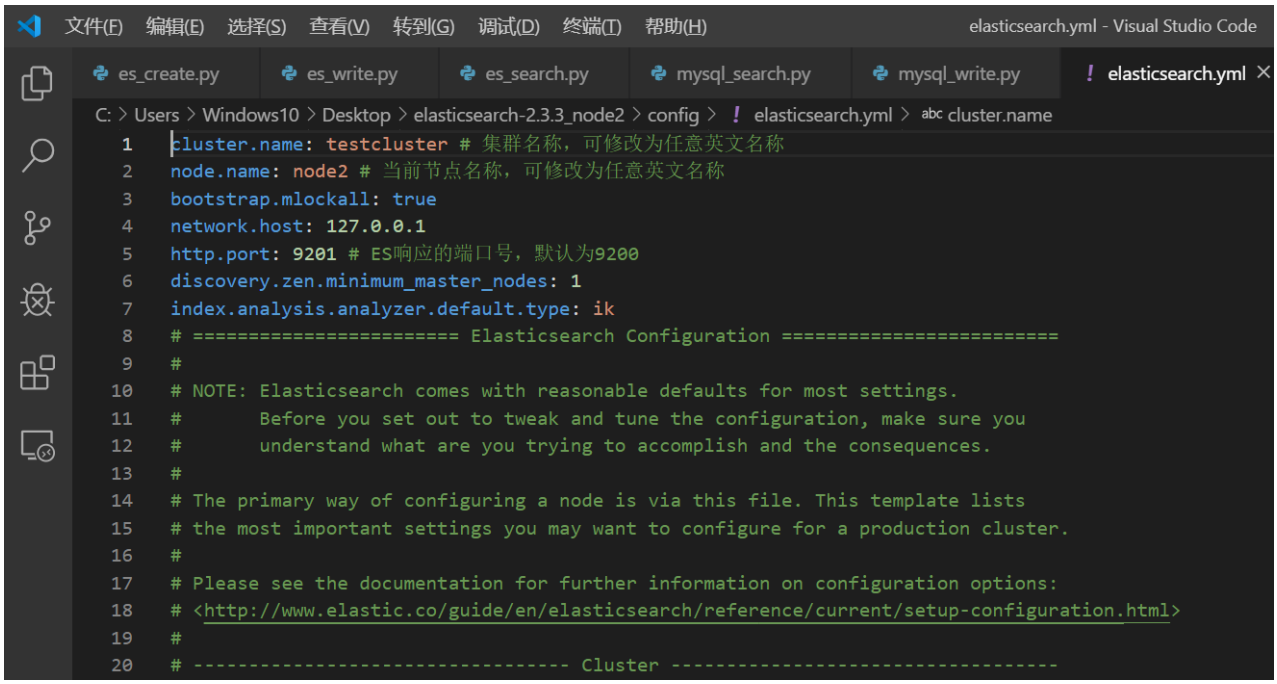


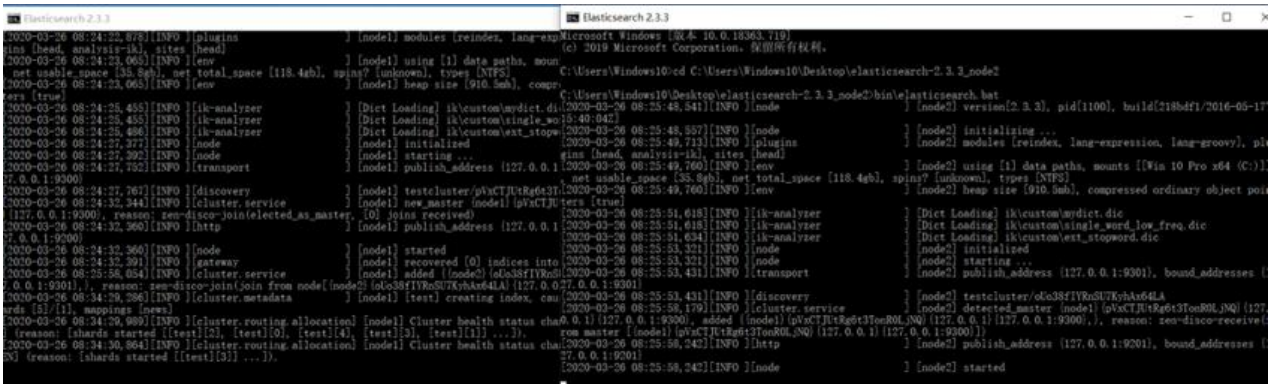
4、启动 ES 数据库服务可视化界面



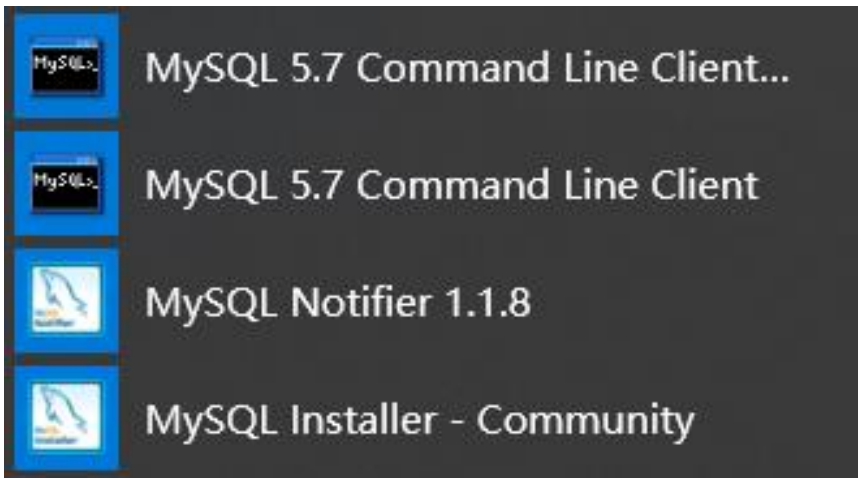


5、构建第二个结点 node2，再次刷新





6、安装 MySQL5.7 并启动 Commond Line Client



7、分别创建 ES 和 MySQL 数据库和表结构

MySQL 5.7 Command Line Client

```
Enter password: ************
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.7.29-log MySQL Community Server (GPL)

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database test;
Query OK, 1 row affected (0.00 sec)

mysql> use test;
Database changed
mysql> CREATE TABLE news (
    ->     id varchar(255) NOT NULL,
    ->     url varchar(255) NOT NULL,
    ->     datestr varchar(255) NOT NULL,
    ->     title varchar(255) NOT NULL,
    ->     content varchar(255) NOT NULL,
    ->     PRIMARY KEY (id)
    -> ) ENGINE=InnoDB DEFAULT CHARSET=utf8;
Query OK, 0 rows affected (0.05 sec)

mysql>
```

C: > Users > Windows10 > Desktop > code > es_create.py > ...

```python
from config import ES_INDEX_NAME, ES_INDEX_DOC_TYPE, es

def create_mapping():
    index_info = {
        'settings':{
            'number_of_replicas': 1,
            'number_of_shards': 5
        },
        'mappings':{
            ES_INDEX_DOC_TYPE:{
                'properties':{
                    "url":{
                        'type': 'string',
                        'index': 'not_analyzed'
                    },
                    "title":{
                        'type': 'string',
                        'index': 'analyzed'
                    },
                    "content":{
                        'type': 'string',
                        'index': 'analyzed'
                    },
                    "date":{
                        'type': 'string',
                        'index': 'not_analyzed'
                    }
                }
            }
        }
    }

```

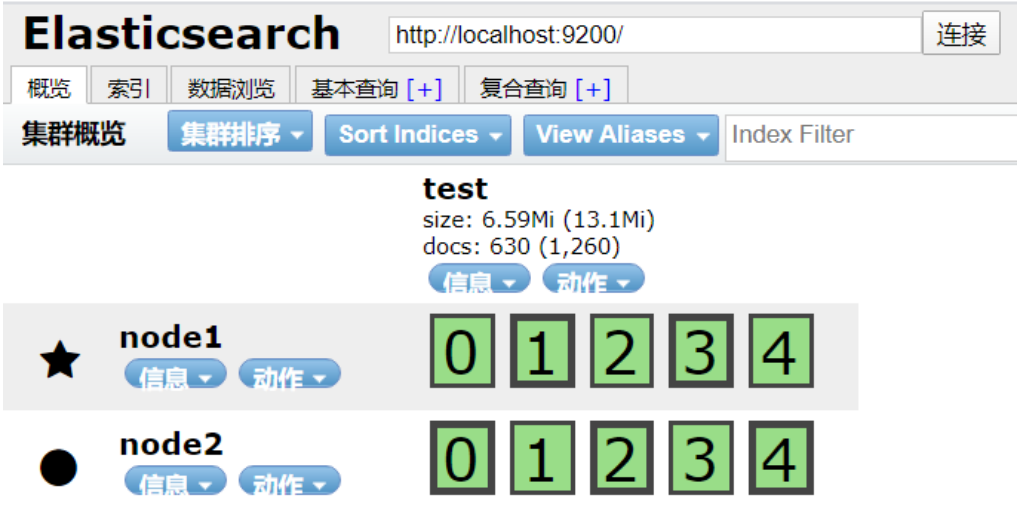
8、执行数据写入

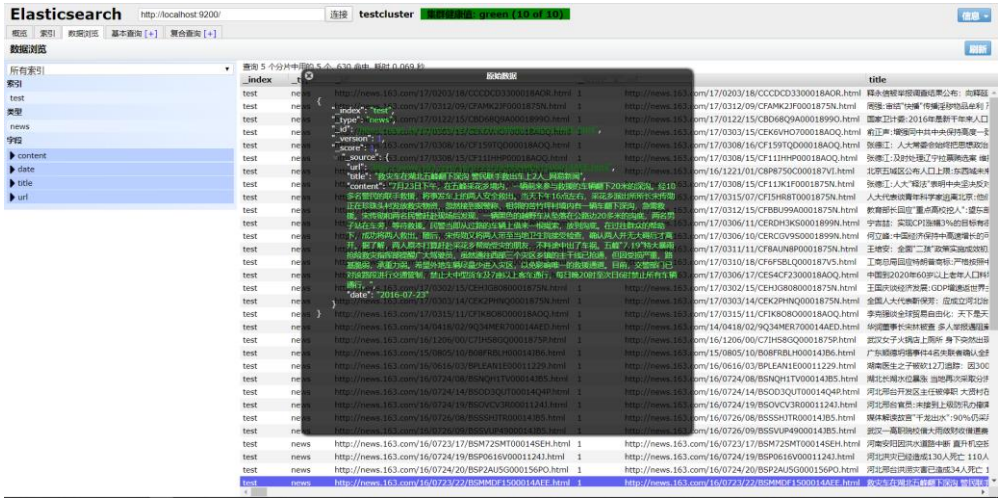
· 使用 ES 写入：

es_create.py | es_write.py | es_search.py | mysql_search.py | mysql_write.py | elasticsearch.yml

C: > Users > Windows10 > Desktop > code > es_write.py > ...

```python
import time
from datetime import datetime

from config import es, ES_INDEX_NAME, ES_INDEX_DOC_TYPE


def write_data(d):
    data = dict()
    data["url"] = d[0]
    data["title"] = d[1]
    data["content"] = d[2]
    data["date"] = d[3]

    es.index(index=ES_INDEX_NAME, doc_type=ES_INDEX_DOC_TYPE, id=data["url"], body=data)


if __name__ == "__main__":
    data_list = []
    with open("news.txt", "r", encoding='UTF-8') as f:
        for line in f:
            url, date, title, content = line.strip().split("|.|")
            data_list.append([url, title, content, date])

    count = 0
    tb = time.time()
    for data in data_list:
        write_data(data)

        count += 1

    print ("ES writing %s items need %s seconds" % (count, time.time() - tb))
```





· 使用 MySQL 写入：

```python
# read data from txt
data_list = []
with open("news.txt", "r", encoding='UTF-8') as f:
    for line in f:
        url, date, title, content = line.strip().split("|.|")
        data_list.append((url, url, title, content, date))

# Connect to the database
connection = pymysql.connect(host=MYSQL_HOST,
                             user=MYSQL_USERNAME,
                             password=MYSQL_PASSWORD,
                             db=MYSQL_DB,
                             charset='utf8mb4',
                             cursorclass=pymysql.cursors.DictCursor)

# write data
count = 0
tb = time.time()
with connection.cursor() as cursor:
    for data in data_list:
        # Create a new record
        sql = "INSERT INTO `" + MYSQL_TABLE + "` (`id`, `url`, `title`, `content`, `datestr`) VALUES (%s, %s, %s, %s, %s)"
        cursor.execute(sql, data)

        # connection is not autocommit by default. So you must commit to save your changes.
        connection.commit()
        count += 1

print ("MySQL writing %s items need %s seconds" % (count, time.time() - tb))
```

· 使用 ES 和 MySQL 进行写入分别耗时 17.2928s 和 1.8589s。

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [版本 10.0.18363.719]
(c) 2019 Microsoft Corporation。保留所有权利。

C:\Users\Windows10>cd C:\Users\Windows10\Desktop\code

C:\Users\Windows10\Desktop\code>python es_write.py
ES writing 630 items need 17.292790174484253 seconds

C:\Users\Windows10\Desktop\code>python mysql_write.py
Traceback (most recent call last):
  File "mysql_write.py", line 32, in <module>
    cursor.execute(sql, data)
  File "D:\anaconda\lib\site-packages\pymysql\cursors.py", line 170, in execute
    result = self._query(query)
  File "D:\anaconda\lib\site-packages\pymysql\cursors.py", line 328, in _query
    conn.query(q)
  File "D:\anaconda\lib\site-packages\pymysql\connections.py", line 517, in query
    self._affected_rows = self._read_query_result(unbuffered=unbuffered)
  File "D:\anaconda\lib\site-packages\pymysql\connections.py", line 732, in _read_query_result
    result.read()
  File "D:\anaconda\lib\site-packages\pymysql\connections.py", line 1075, in read
    first_packet = self.connection._read_packet()
  File "D:\anaconda\lib\site-packages\pymysql\connections.py", line 684, in _read_packet
    packet.check_error()
  File "D:\anaconda\lib\site-packages\pymysql\protocol.py", line 220, in check_error
    err.raise_mysql_exception(self._data)
  File "D:\anaconda\lib\site-packages\pymysql\err.py", line 109, in raise_mysql_exception
    raise errorclass(errno, errval)
pymysql.err.DataError: (1406, "Data too long for column 'content' at row 1")

C:\Users\Windows10\Desktop\code>python mysql_write.py
D:\anaconda\lib\site-packages\pymysql\cursors.py:329: Warning: (1265, "Data truncated for column 'content' at row 1")
  self._do_get_result()
MySQL writing 630 items need 1.8588917255401611 seconds

C:\Users\Windows10\Desktop\code>
```

9、执行全文搜索

· 使用 ES 执行搜索代码：

```python
import json
import time
from config import es, ES_INDEX_NAME, ES_INDEX_DOC_TYPE


def query_by_keyword():
    query_body = {
        "query":{
            "match": { "title": "中国"} # 新闻标题中包含中国关键词
        },
        "size": 10000
    }

    tb = time.time()
    results = es.search(index=ES_INDEX_NAME, doc_type=ES_INDEX_DOC_TYPE, body=query_body)["hits"]["hits"]

    ff = open("search_results.txt", "w", encoding="UTF-8")
    for item in results:
        data = item["_source"]
        ff.write("%s|.|%s|.|%s|.|%s\n" % (data["url"], data["date"], data["title"], data["content"]))
    ff.close()
    tb1 = time.time()
    print ("ES search data time (seconds): ", tb1 - tb)


if __name__ == "__main__":
    query_by_keyword()
```

· 使用 MySQL 执行搜索代码：

```python
import time
import pymysql.cursors

from config import MYSQL_HOST, MYSQL_USERNAME, MYSQL_PASSWORD, MYSQL_DB, MYSQL_TABLE


# Connect to the database
connection = pymysql.connect(host=MYSQL_HOST,
                             user=MYSQL_USERNAME,
                             password=MYSQL_PASSWORD,
                             db=MYSQL_DB,
                             charset='utf8mb4',
                             cursorclass=pymysql.cursors.DictCursor)

tb = time.time()

with connection.cursor() as cursor:
    # Read a single record
    sql = "SELECT * FROM `" + MYSQL_TABLE + "` WHERE `title` like %s"
    cursor.execute(sql, ("%中国%",))
    result = cursor.fetchall()

    ff = open("search_results.txt", "w", encoding="UTF-8")
    for r in result:
        ff.write("%s|.|%s|.|%s|.|%s\n" % (r["url"], r["datestr"], r["title"], r["content"]))
    ff.close()
tb1 = time.time()
print ("MySQL search data time (seconds): ", tb1 - tb)
```

search_results - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

http://news.163.com/17/0530/15/CLMOCL3K0001875O.html|.|2017-05-30|.|英媒质疑"中国造"蒙内铁路 肯尼亚人:是你们无知_网易新闻|.|由中国路桥承建的蒙内铁路，全长约480公里，总投资38亿美元，是肯尼亚独立以来修建的最大基建项目。百年前，英国殖民者花费550万英镑，在东非主持修筑一条铁路，因为修筑艰难等原因，被称为“疯狂铁路”。百年后，中国承建的蒙内铁路与之蒙巴萨-内罗毕段米轨铁路基本并行，引发一些英国媒体质疑：这不过是另一条“疯狂铁路”。事实果真如此吗？肯尼亚：铁轨上的国家“中国造”蒙巴萨-内罗毕标轨铁路即将开通的消息，让肯尼亚这个东非国家兴奋不已。在肯尼亚内罗毕的铁路博物馆里，退役火车头和历史老照片讲述着肯尼亚第一条铁路的故事。“肯尼亚是诞生在铁轨上的国家。”博物馆的小册子如此介绍。米轨铁路的轨距为1米，属于窄轨铁路。百年前，“肯尼亚”这个名字尚不存在，对于英国殖民者而言，这里只是一片无名荒原，横亘在他们和乌干达丰富的矿产资源之间。为打开乌干达大门，加强对“英属东非”的控制，也为了遏制当时已臭名昭著的奴隶贸易，英国政府决定在东非修筑一条铁路，连接印度洋上的蒙巴萨港和内陆的乌干达，并命名其为“乌干达铁路”。对于不了解非洲的英国政客来说，斥巨资在“黑色大陆”上修铁路的决定荒唐可笑，一位英国议员甚至赋诗讽刺其是“疯狂铁路”（lunatic line）。疯狂铁路如今，这条931公里铁路仍担有此名，原因之一是当年的修建艰难异常，超乎想象。“当时修建铁路的印度劳工没有现在的大型机械，很多铺设工作只能依靠简单的工具，甚至是徒手。”铁路博物馆馆长巴拉萨介绍说。当地部族把铁路称为“铁蛇”，不满“铁蛇”入侵拿起武器反抗，也使得修建举步维艰。根据博物馆的资料显示：从1896年修建开始到1901年铁路竣工，共有2493名铁路工人死亡。他们多是死于当地部落袭击，疟疾、痢疾等传染病，以及令人闻风丧胆的“察沃的食人狮”。狮子多次袭击铁路工人、制造恐慌，甚至在1900年6月将正在巡查铁路的英国官员拖出车厢咬死。“每一英里米轨铁路，都是由四条铁路工人的生命铺就。”独立之路对后来乘着火车欣赏殖民地草原美景的英国贵妇和探险家们来说，这恐怕只是下午茶时的八卦谈资，当时一幅“乌干达铁路游”广告甚至画着一位火车乘客拍手逗弄草原上的狮子。英国殖民者恐怕不曾料到，这条耗费巨大人力物力修建的铁路，却最终帮助肯尼亚走向独立。“当时的许多爱国人士就是乘着火车，到肯尼亚四处演说，为独立运动争取支持。一些武器也通过铁路运送到了反抗军的手里。”巴拉萨说。一条英国殖民者修建的铁路，却最终帮助终结了殖民统治，这或许是肯尼亚的米轨铁路给历史留下的最大吊诡。同时，铁路所经之处大农场纷纷兴起。肯尼亚的咖啡和牛肉借助铁路运输走向海外，经济的发展使得肯尼亚更有底气对英国统治者说不。铁路的修建还带动了城市的形成。如今的东非大都市内罗毕，便是起源于铁路工人与当地马赛人以物换物的市场。新铁路 新希望而一百年前乘着火车奔向独立的肯尼亚，如今期待另一条铁路助其实现经济腾飞：米轨铁路如今已运营超过百年，自然老化和缺乏维护使得其速度从最初设计的每小时70公里下降到了40公里，肯尼亚大部分的货物运输也因此转移到了公路上。同许多肯尼亚人一样，巴拉萨如今期待设计客运时速120公里（货运80公里）的蒙内铁路能重新激活肯尼亚的铁路系统，并推动肯尼亚的工业化。“未来乘坐蒙内铁路客车可以在一天之内往返内罗毕和蒙巴萨，这是以前坐火车和大巴都无法实现的。“货物能在一天之内从蒙巴萨抵达内罗毕，而靠公路运输则可能要好几天。”巴拉萨说，随着人员与货物流动成本的降低，沿线的商业和工业都将获益。绝对不是“疯狂铁路”而在巴拉萨眼里，米轨铁路曾为肯尼亚带来了了不起的变化，“疯狂铁路”这个称呼只能说明欧洲殖民者对非洲的无知和轻视。修建标轨铁路更是肯尼亚做出的一个“明智选择”，他说，中国公司出色地完成了建造任务，也向世界证明了这一点。“蒙内铁路绝不是另一条疯狂的铁路，这是一条清醒而明智的铁路。”巴拉萨非常肯定地说。

http://news.163.com/17/0223/00/CDTUAFAQ000187V5.html|.|2017-02-23|.|37岁姚明将任中国篮协主席? 年轻"掌门"非个例_网

· 使用 ES 和 MySQL 进行搜索分别耗时 2.2339s 和 0.0314s

```
C:\WINDOWS\system32\cmd.exe
ES writing 630 items need 17.292790174484253 seconds

C:\Users\Windows10\Desktop\code>python mysql_write.py
Traceback (most recent call last):
  File "mysql_write.py", line 32, in <module>
    cursor.execute(sql, data)
  File "D:\anaconda\lib\site-packages\pymysql\cursors.py", line 170, in execute
    result = self._query(query)
  File "D:\anaconda\lib\site-packages\pymysql\cursors.py", line 328, in _query
    conn.query(q)
  File "D:\anaconda\lib\site-packages\pymysql\connections.py", line 517, in query
    self._affected_rows = self._read_query_result(unbuffered=unbuffered)
  File "D:\anaconda\lib\site-packages\pymysql\connections.py", line 732, in _read_query_result
    result.read()
  File "D:\anaconda\lib\site-packages\pymysql\connections.py", line 1075, in read
    first_packet = self.connection._read_packet()
  File "D:\anaconda\lib\site-packages\pymysql\connections.py", line 684, in _read_packet
    packet.check_error()
  File "D:\anaconda\lib\site-packages\pymysql\protocol.py", line 220, in check_error
    err.raise_mysql_exception(self._data)
  File "D:\anaconda\lib\site-packages\pymysql\err.py", line 109, in raise_mysql_exception
    raise errorclass(errno, errval)
pymysql.err.DataError: (1406, "Data too long for column 'content' at row 1")

C:\Users\Windows10\Desktop\code>python mysql_write.py
D:\anaconda\lib\site-packages\pymysql\cursors.py:329: Warning: (1265, "Data truncated for column 'content' at row 1")
  self._do_get_result()
MySQL writing 630 items need 1.8588917255401611 seconds

C:\Users\Windows10\Desktop\code>python es_search.py
ES search data time (seconds):  2.2338857650756836

C:\Users\Windows10\Desktop\code>python mysql_search.py
MySQL search data time (seconds):  0.03143429756164551

C:\Users\Windows10\Desktop\code>
```

结论：Elasticsearch 与 MySQL 比较，在小规模数据情况下，MySQL 的写入与搜索速度更快。