**基于大数据分析的移动音乐智能搜索与推荐平台**

**部署文档**

目录

[1 依赖软件列表介绍 2](#_Toc29289433)

[2 服务器需求 2](#_Toc29289434)

[3 组件安装 2](#_Toc29289435)

[3.1 Neo4J安装（知识图谱——图数据库） 2](#_Toc29289436)

[3.2 知识图谱更新（知识图谱——构建） 4](#_Toc29289437)

[3.2.1 爬虫程序（2个程序） 4](#_Toc29289438)

[3.2.2 知识图谱构建（信息抽取+知识融合+neo4j更新）（2个程序） 9](#_Toc29289439)

[3.2.3 音频标签标注（1个程序） 15](#_Toc29289440)

[3.2.4 歌词标签标注+集成标签预测+人工标签修改（3个程序） 19](#_Toc29289441)

[3.3 SolrCloud安装 25](#_Toc29289442)

[3.3.1 zookeeper安装 25](#_Toc29289443)

[3.3.2 solr安装及配置 26](#_Toc29289444)

[3.3.3 添加collection 28](#_Toc29289445)

[3.4 索引更新子系统安装 31](#_Toc29289446)

[4 集群设计 31](#_Toc29289447)

[4.1 neo4j集群 31](#_Toc29289448)

[5 对话系统部署 31](#_Toc29289449)

[5.1 部署路径 31](#_Toc29289450)

[5.2 程序安装 32](#_Toc29289451)

[5.3 验证服务 35](#_Toc29289452)

[5.4 日志文件 35](#_Toc29289453)

[5.5 端口信息 35](#_Toc29289454)

[5.6 程序停止 35](#_Toc29289455)

[6 QA 35](#_Toc29289456)

[6.1 部署问题及解决办法 35](#_Toc29289457)

# 依赖软件列表介绍

《基于大数据分析的移动音乐智能搜索与推荐平台》项目涉及的第三方软件包括：

Neo4J, Solr, MySQL, Redis

涉及安装包包括：

solr-7.7.1.tar.gz

neo4j-community-3.4.0-unix.tar.gz（neo4j程序）

zookeeper 3.4.14

proxy\_pool.tar.gz（开源代理ip池）

redis-5.0.5.tar.gz

# 服务器需求

服务部署的相关详细信息如下：

|  |  |  |
| --- | --- | --- |
| 服务名称 | 服务器 | 服务器数量 |
| Neo4J集群 | **172.18.28.4、172.18.28.5、172.18.28.6** | 3台 |
| 爬虫服务器 | **172.18.28.11** | 1台 |
| 知识图谱构建 | **172.18.28.9** | 1台 |
| 音频标签预测 | **172.18.28.8** | 1台 |
| 歌词标签预测+集成标签预测 | **172.18.28.7** | 1台 |
| SolrCloud集群 | **172.18.28.13-21** | 公司现有 |
| 对话系统 | **172.18.28.10** | 1台 |
| Zookeeper集群 | **172.18.254.52-56** | 公司现有 |
| Mysql服务器 | **172.17.104.16** | 公司现有 |
| Elasticsearch服务器 |  | 公司现有 |
| Redis 集群 |  | 公司现有 |

# 组件安装

## 3.1 Neo4J安装（知识图谱——图数据库）

* **地址：**

172.18.28.4

172.18.28.5

172.18.28.6

三台都要安装

* **资源:**

neo4j压缩包:

<http://172.21.151.6:82/svn/java/maven/quku/iMusicSong_PLF_1.0/scut/scut-MSDS/trunk/kg/resource/neo4j-community-3.4.0.tar.gz>

* **依赖环境（java8）**

# 切换root，安装java8

yum install -y java-1.8.0-openjdk-devel.x86\_64

* **安装neo4j**

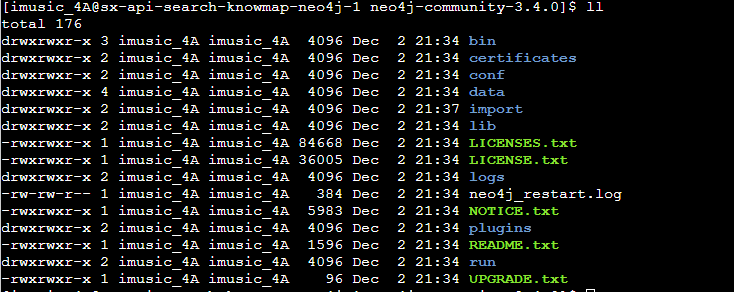
1. 将neo4j压缩包复制到/backup目录下并解压

cd /backup

tar -zxvf neo4j-community-3.4.0.tar.gz

cd neo4j-community-3.4.0-unix

此为neo4j的目录结构：



1. 修改配置：

# 配置文件为conf/neo4j.conf

vi conf/neo4j.conf

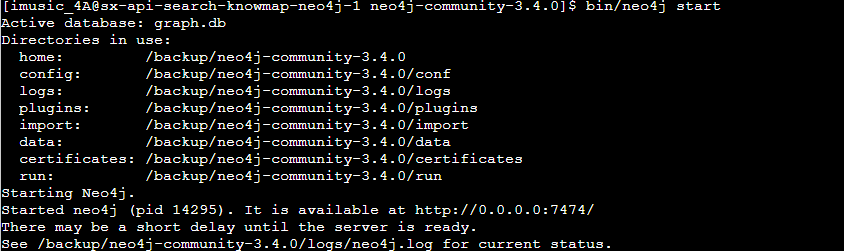
一般来说使用安装包里面的配置文件即可，已经是配置过的。

值得注意的配置有以下：

|  |
| --- |
| # 初始堆大小  dbms.memory.heap.initial\_size=6g  # 最大堆大小  dbms.memory.heap.max\_size=6g  # 页面缓存大小  dbms.memory.pagecache.size=7g  # bolt协议端口  dbms.connector.bolt.listen\_address=:8080 # http协议端口  dbms.connector.http.listen\_address=:7474  # 事务日志保存设置  dbms.checkpoint.interval.time=30s  dbms.checkpoint.interval.tx=1  dbms.tx\_log.rotation.retention\_policy=false  dbms.tx\_log.rotation.size=1M  # 使用apoc插件  dbms.security.procedures.unrestricted=apoc.\* |

1. 启动服务

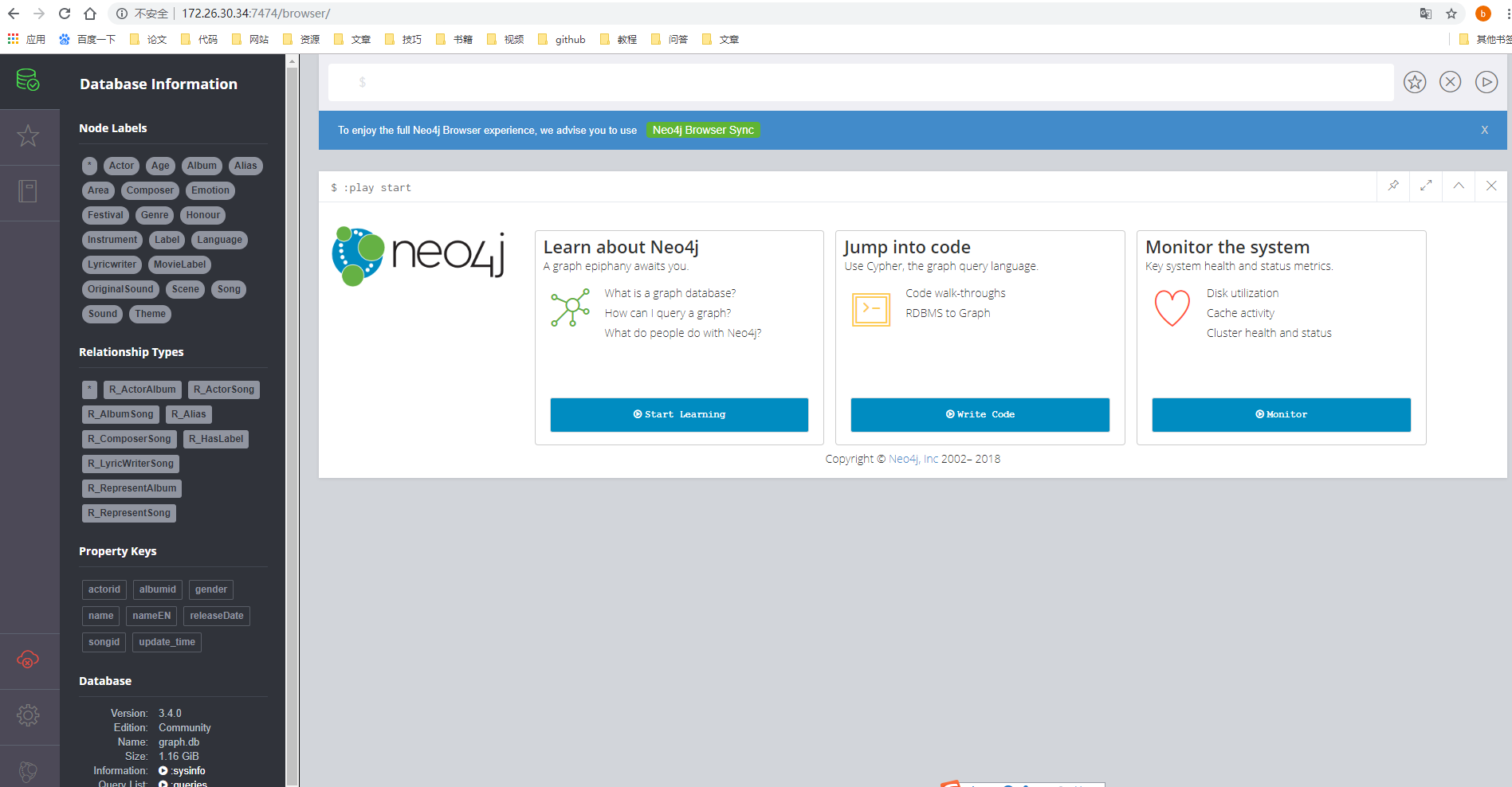
bin/neo4j start



# bin/neo4j stop为停止服务，bin/neo4j restart为重启服务

* **验证服务**

启动服务过几分钟之后，浏览器访问<http://172.26.30.34:7474/>，出现以下画面即为启动成功



* **定时监控服务**

crontab -e

|  |
| --- |
| \*/1 \* \* \* \* sh /backup/restart\_db.sh |

# 每隔一分钟检查是否有neo4j进程，没有则重启

## 3.2 知识图谱更新（知识图谱——构建）

# 所有步骤都要先把/backup目录权限开启，后面不再赘述

chmod 777 /backup

### 3.2.1 爬虫程序（2个程序）

1. **地址**

172.18.28.11

1. **代码及资源地址**

代码1【代理ip池】:

<http://172.21.151.6:82/svn/java/maven/quku/iMusicSong_PLF_1.0/scut/scut-MSDS/trunk/kg/src/proxy_pool/>

代码2【爬虫程序】:

<http://172.21.151.6:82/svn/java/maven/quku/iMusicSong_PLF_1.0/scut/scut-MSDS/trunk/kg/src/scut_crawler/>

资源1【pyhon3.6源代码】:

<http://172.21.151.6:82/svn/java/maven/quku/iMusicSong_PLF_1.0/scut/scut-MSDS/trunk/kg/resource/>Python-3.6.6.tgz

1. **安装python环境**
2. 因为爬虫程序中一个加解密模块有用到c++库，所以本机器的python版本需要安装源码编译的python，【资源1】为python3.6源代码，将其移动到/backup路径下并切换到/backup路径下

cd /backup

1. 切换root，安装c++依赖

yum install openssl-devel

yum install zlib-devel

yum install bzip2-devel

1. 切换imusic\_4A，解压并进入目录

tar -zxvf Python-3.6.6.tgz

cd /backup/Python-3.6.6

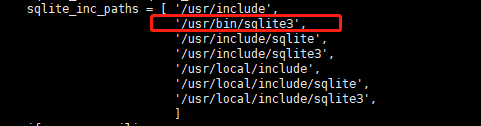
1. 配置编译路径

./configure -prefix=/backup/Python366

1. 配置sqlite3模块路径

vi setup.py

# 找到本机的sqlite路径并配置到其中的sqlite\_inc\_paths中



# 如上所示，本机的sqlite3的路径为/usr/bin/sqlite3

1. 编译安装

make && make install

1. 在/backup路径创建虚拟环境

cd /backup

Python366/bin/python3 -m venv crawler

1. **更改pip源，提高pip装库速度**
2. mkdir ~/.pip
3. touch ~/.pip/pip.conf
4. vi ~/.pip/pip.conf

# 更换成清华源，输入以下内容

|  |
| --- |
| [global]  index-url = <https://pypi.tuna.tsinghua.edu.cn/simple> |

1. **部署程序1【代理ip池】**

功能简介：为爬虫程序提供代理ip

此程序为第三方程序，来源：<https://github.com/jhao104/proxy_pool>

1. 将【代码1】【代理ip池】移动到/backup路径下并切换到代码目录

cd /backup/proxy\_pool

1. 安装代理ip池所需python库

source /backup/crawler/bin/activate

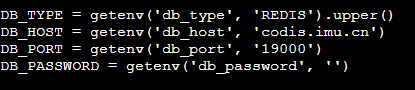
pip install -r requirements.txt

1. 修改redis端口配置

vi Config/setting.py

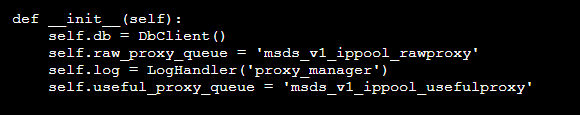
# 修改redis配置

|  |
| --- |
| DB\_TYPE = getenv('db\_type', 'REDIS').upper()  DB\_HOST = getenv('db\_host', 'codis.imu.cn')  DB\_PORT = getenv('db\_port', '19000')  DB\_PASSWORD = getenv('db\_password', '') |



1. 修改存放在redis的键值对名字,改成公司要求的命名方式

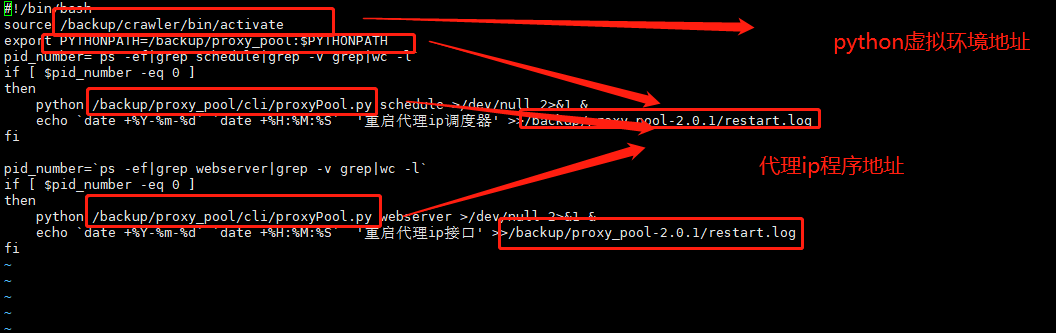
vi Manager/ProxyManager.py



1. 运行脚本

vi restart\_proxypool.sh

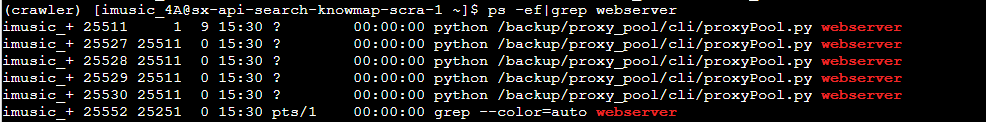
# 修改项目路径



sh restart\_proxypool.sh

1. 验证代理服务是否正常启动

ps -ef|grep webserver



ps -ef|grep schedule



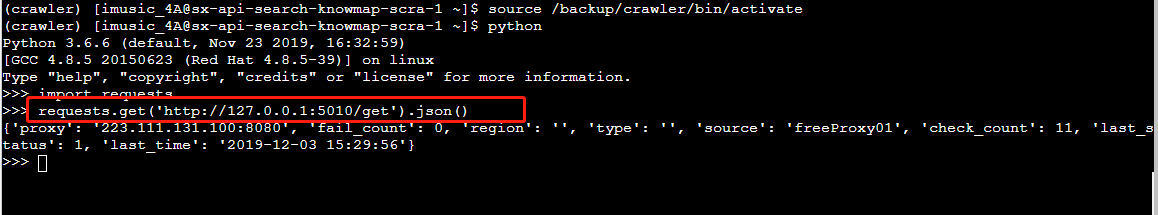
1. 访问端口获取代理ip（端口默认为5010，可在Config/settings.py中进行修改）

# 过几分钟后，验证代理ip是否获取成功，若如下所示则成功，proxy中的内容即为免费代理ip，check\_count为检测次数，last\_time为上次检测时间，别台服务器上若需要访问则需在防火墙打开该端口

python

>>> import requests

>>> requests.get('http://127.0.0.1:5010/get').json()



1. 制定定时任务

crontab -e

# 加入定时任务，每分钟运行一次脚本，如果不存在webserver和schedule两个进程，则重启

|  |
| --- |
| \*/1 \* \* \* \* sh /backup/proxy\_pool/restart\_proxypool.sh |

1. **部署程序2【爬虫】**

功能简介：爬取歌曲和歌单

1. 将【代码2】【爬虫】复制到/backup路径下并切换到代码目录

cd /backup/scut\_crawler/labelcrawler

1. 安装爬虫必备的库

source /backup/crawler/bin/activate

pip install -r requirements.txt

1. 配置数据库连接

vi /backup/scut\_crawler/labelcrawler/settings.py

# 修改数据库配置、es配置和代理ip配置

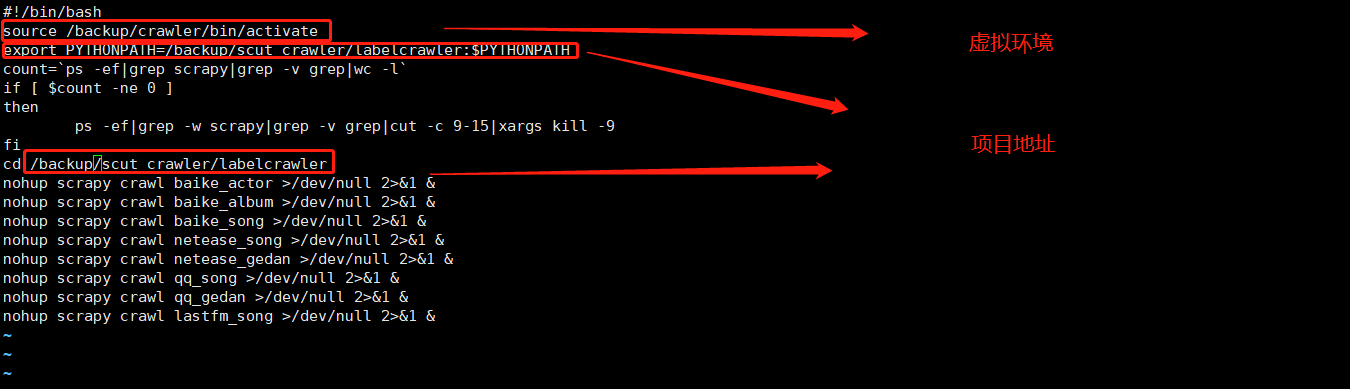
|  |
| --- |
| # db  DB\_HOST = '172.17.104.16'  DB\_USER = 'read\_song'  DB\_PASSWORD = 'Ss#u3w#H8'  DB\_PORT = '3307'  DB\_DATABASE = 'musiclib'  DB\_ACTOR = 'search\_content'  DB\_ALBUM = 'search\_content'  DB\_SONG = 'search\_content'  DB\_TAG = 'multimodal\_tag'  CRAWL\_BAIKE\_ACTOR = 'crawl\_baike\_actorid'  CRAWL\_BAIKE\_SONG = 'crawl\_baike\_songid'  CRAWL\_BAIKE\_ALBUM = 'crawl\_baike\_albumid'  CRAWL\_QQ\_SONG = 'crawl\_qq\_songid'  CRAWL\_QQ\_GEDAN = 'crawl\_qq\_gedanid'  CRAWL\_NETEASE\_SONG = 'crawl\_netease\_songid'  CRAWL\_NETEASE\_GEDAN = 'crawl\_netease\_gedanid'  # es  ES\_CLIENT = 'http://esdata.imu.cn'  ES\_PORT = '8200'  ES\_USER = 'scutnetwork'  ES\_PASSWORD = 'scut7872=kaJIW'  ES\_BAIKE\_ACTOR = 'scut\_baike\_actor'  ES\_BAIKE\_ALBUM = 'scut\_baike\_album'  ES\_BAIKE\_SONG = 'scut\_baike\_song'  ES\_QQ\_SONG = 'scut\_qq\_song'  ES\_QQ\_SEARCH\_GEDAN = 'scut\_qq\_search\_gedan\_tag'  ES\_NETEASE\_SONG = 'scut\_netease\_song'  # proxy\_ip  PROXY\_IP = 'http://127.0.0.1:5010/get/' |



1. 启动爬虫

vi run\_crawler.sh

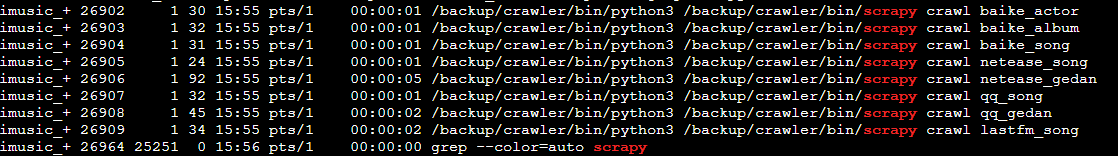
# 修改项目路径



sh /backup/scut\_crawler/run\_crawler.sh

1. 验证服务

ps -ef|grep scrapy



1. 制定定时任务

crontab -e

# 加入定时任务，每天晚上11点开始启动，启动之前会先kill掉上一次的爬虫

|  |
| --- |
| 0 23 \* \* \* sh /backup/scut\_crawler/run\_crawler.sh |

### 3.2.2 知识图谱构建（信息抽取+知识融合+neo4j更新）（2个程序）

1. **地址**

172.18.28.9

1. **代码及资源地址**

代码:

<http://172.21.151.6:82/svn/java/maven/quku/iMusicSong_PLF_1.0/scut/scut-MSDS/trunk/kg/src/build_knowledge/>

1. **安装python(yum安装)**
2. 切换root

yum install python36

1. 切换成普通用户imusic\_4A，在/backup路径创建虚拟环境kgbuild

python3 -m venv kgbuild

1. 配置pip源

mkdir ~/.pip

touch ~/.pip/pip.conf

vi ~/.pip/pip.conf

# 更换成清华源，输入以下内容

|  |
| --- |
| [global]  index-url = <https://pypi.tuna.tsinghua.edu.cn/simple> |

1. **配置并调试程序1【信息抽取+知识融合】**

功能简介：将百科上爬取到的音乐知识进行信息抽取，然后对齐到本地曲库id，最后经过规范化处理后存到mysql数据库中。

1. 将【代码】复制到/backup目录下，并进入代码目录

cd /backup/build\_knowledge

1. 安装程序必须的python库

source /backup/kgbuild/bin/activate

pip install -r requirements.txt

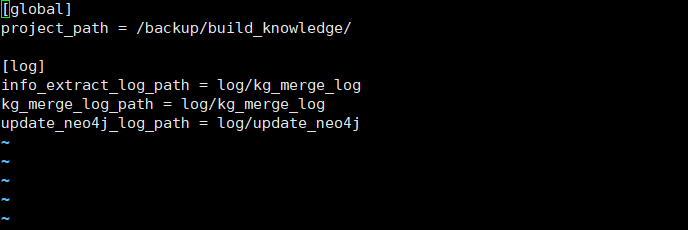
1. 配置项目路径

vi config/global\_conf.ini

# 修改project\_path为本项目路径

# 注意最后不能少了斜杠

|  |
| --- |
| [global]  project\_path = /backup/build\_knowledge/  [log]  info\_extract\_log\_path = log/kg\_merge\_log  kg\_merge\_log\_path = log/kg\_merge\_log  update\_neo4j\_log\_path = log/update\_neo4j |

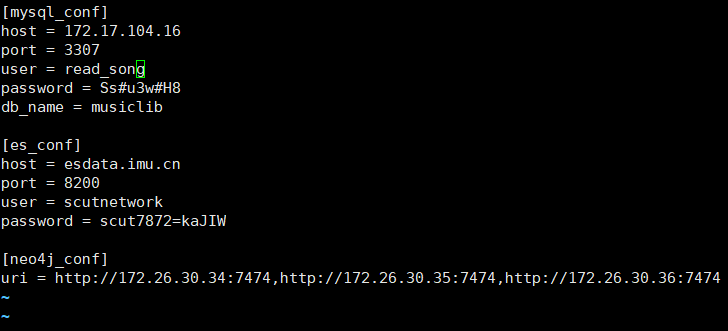


1. 配置数据库连接

vi config/db\_conf.ini

# 修改数据库和es配置信息

|  |
| --- |
| [mysql\_conf]  host = 172.17.104.16  port = 3307  user = read\_song  password = Ss#u3w#H8  db\_name = musiclib  [es\_conf]  host = esdata.imu.cn  port = 8200  user = scutnetwork  password = scut7872=kaJIW  [neo4j\_conf]  uri = http://172.26.30.34:7474,http://172.26.30.35:7474,http://172.26.30.36:7474 |

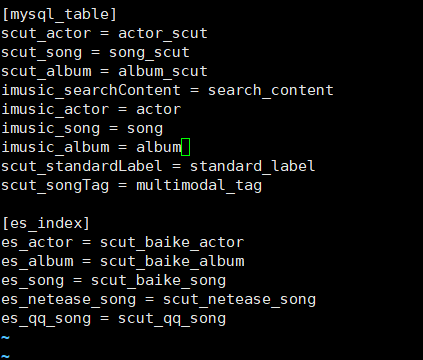


1. 配置数据表

vi config/table.ini

# 修改数据库表和es的index

|  |
| --- |
| [mysql\_table]  scut\_actor = actor\_scut  scut\_song = song\_scut  scut\_album = album\_scut  imusic\_searchContent = search\_content  imusic\_actor = actor  imusic\_song = song  imusic\_album = album  scut\_standardLabel = standard\_label  scut\_songTag = multimodal\_tag  [es\_index]  es\_actor = scut\_baike\_actor  es\_album = scut\_baike\_album  es\_song = scut\_baike\_song  es\_netease\_song = scut\_netease\_song  es\_qq\_song = scut\_qq\_song |



1. 调试程序1【信息抽取+知识融合】

# 本程序与程序2【neo4j更新】程序可能产生冲突，为避免发生，尽量不要让这两个程序同时运行

# 激活虚拟环境

source /backup/kgbuild/bin/activate

# 更改环境变量

export PYTHONPATH=/backup/build\_knowledge:$PYTHONPATH

# 判断系统中是否有程序2【neo4j更新】在运行，如果有，请先kill掉

ps -ef|grep update\_neo4j\_main.py

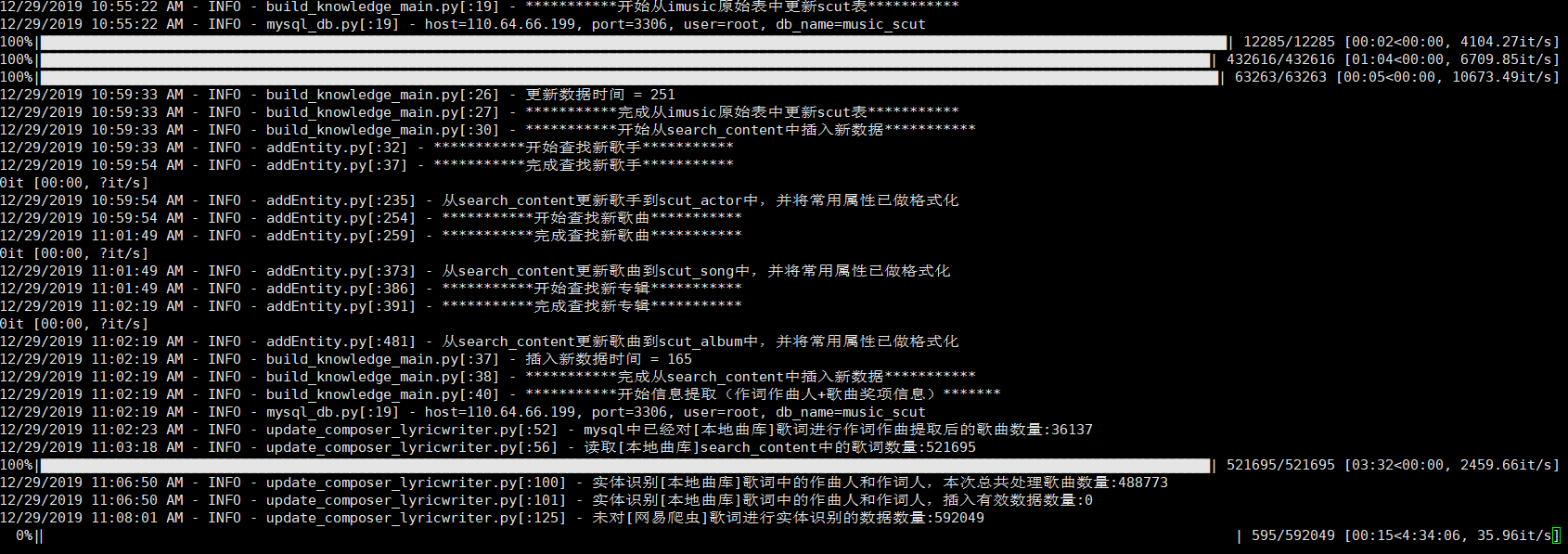
# 判断系统中是否已经有程序1【信息抽取+知识融合】在运行，如果有，请先kill掉

ps -ef|grep build\_knowledge\_main.py

# 运行程序

python /backup/build\_knowledge/build\_knowledge\_main.py

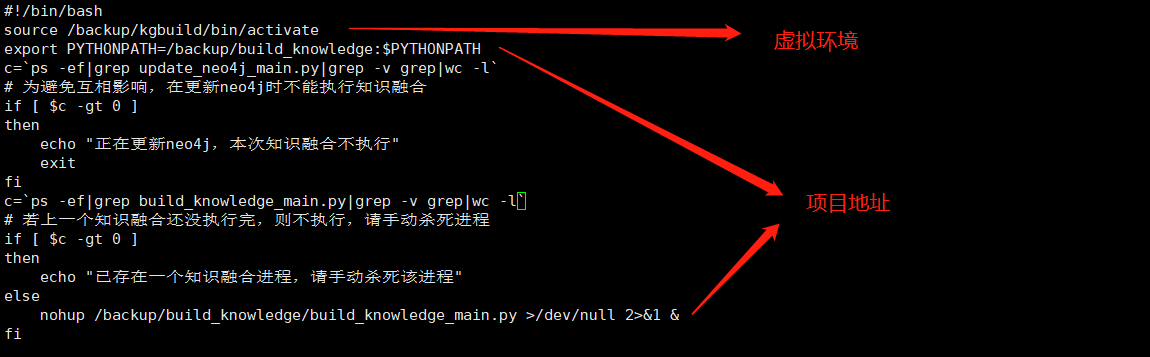
# 看到如下信息并且无报错，则证明配置成功了，程序运行时间可能比较久，可以先kill掉，为其设置定时任务



1. 配置脚本，制定定时任务

# 本程序与程序2【更新neo4j】可能产生冲突，为避免发生，因程序1【更新neo4j】优先级更高，当程序2【更新neo4j】运行时，本程序不会运行；当本程序上一个周期的进程还未执行完，本程序也不会执行

vi build.sh



crontab -e

# 加入定时任务，每天中午12点开始运行

|  |
| --- |
| 00 12 \* \* \* sh /backup/build\_knowledge/build.sh |

1. 调试程序2【neo4j更新】程序

功能简介:根据search\_content中的歌曲，查找mysql数据库中对应的格式化好的各类数据，更新到neo4j中。

# 本程序与程序1【信息抽取+知识融合】程序可能产生冲突，为避免发生，尽量不要让这两个程序同时运行

# 激活虚拟环境

source /backup/kgbuild/bin/activate

# 修改环境变量

export PYTHONPATH=/backup/build\_knowledge:$PYTHONPATH

# 判断系统中是否已经有程序2【neo4j更新】在运行，如果有，请先kill掉

ps -ef|grep update\_neo4j\_main.py

# 判断系统中是否有程序1【信息抽取+知识融合】在运行，如果有，请先kill掉

ps -ef|grep build\_knowledge\_main.py

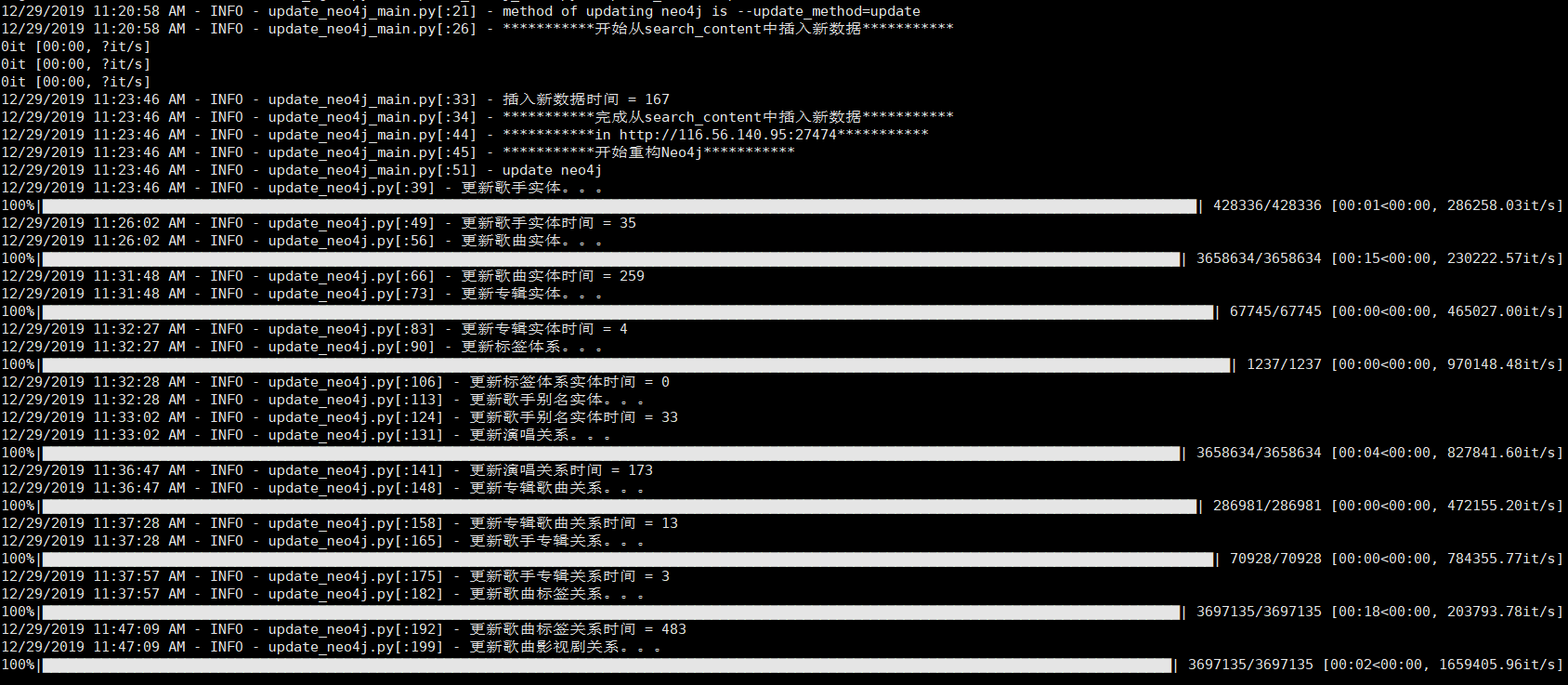
# 如果neo4j是刚解压的，什么东西都没有，第一次构建知识图谱请使用如下命令

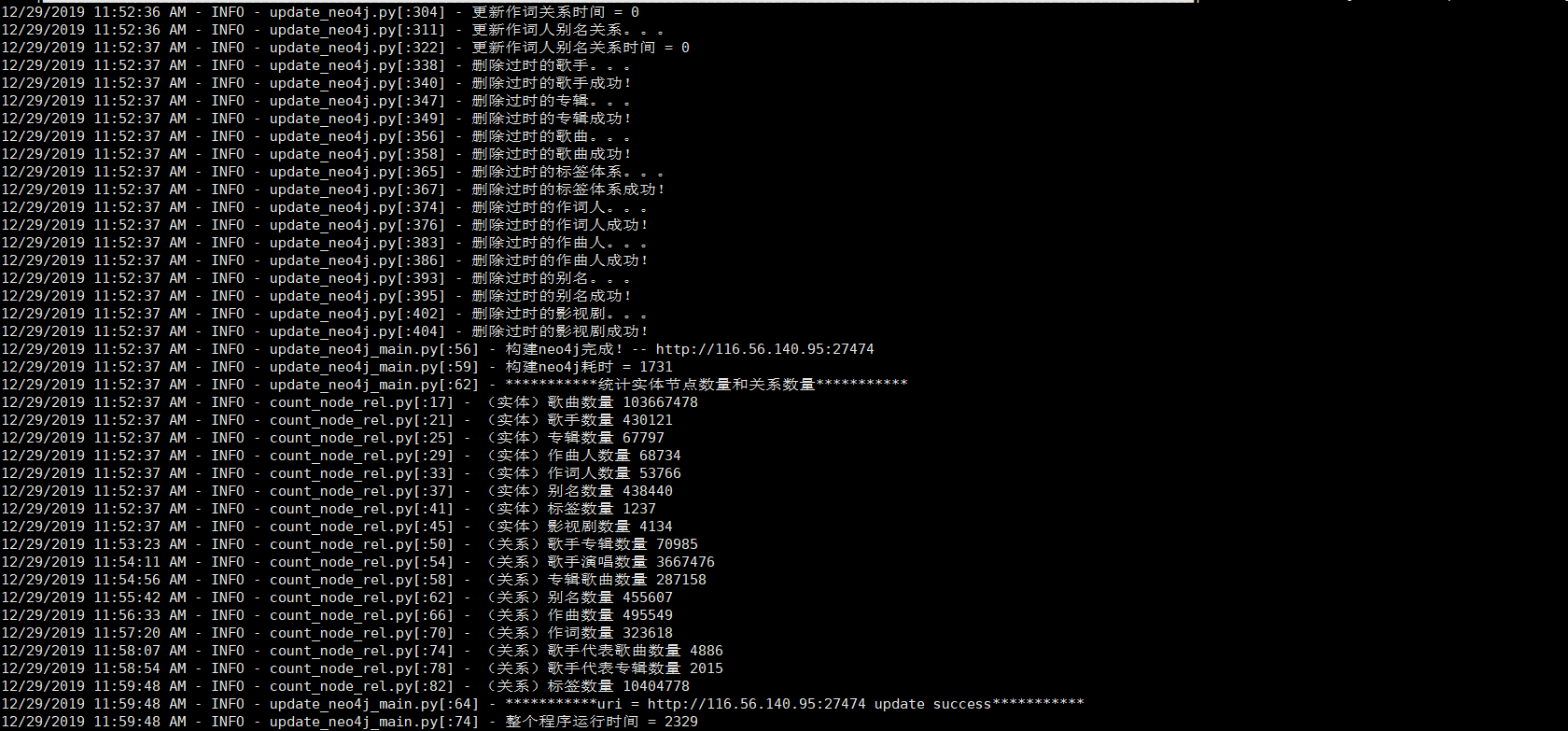
python /backup/build\_knowledge/update\_neo4j\_main.py --update\_method=rebuild

# 如果neo4j里面已经有数据，那么后期的更新请使用如下命令

python /backup/build\_knowledge/update\_neo4j\_main.py --update\_method=update

# 看到如下信息并且无报错，则证明配置成功了，程序运行时间可能比较久，可以先kill掉，为其设置定时任务

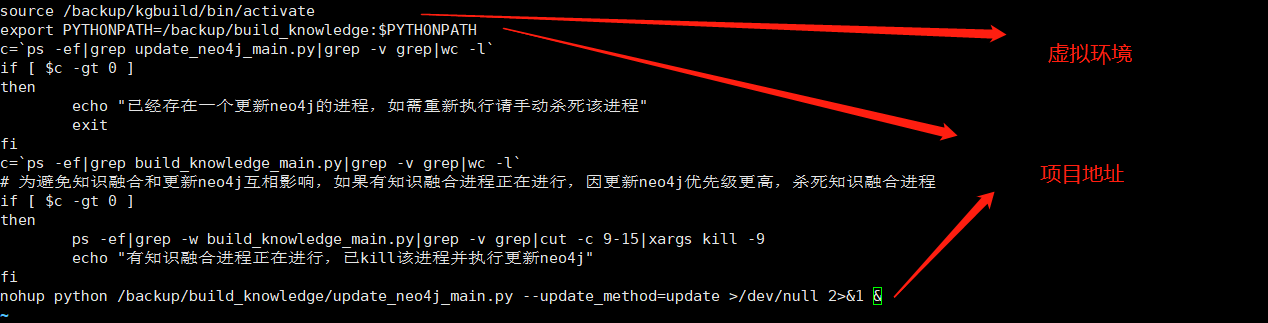




1. 配置脚本，制定定时任务

# 本程序与程序1【信息抽取+知识融合】程序可能产生冲突，为避免发生，当知识融合程序运行时，本程序会将其kill掉；当本程序上一个周期的进程还未执行完，本程序也不会执行

vi update.sh



crontab -e

# 加入定时任务，每天0点更新neo4j

|  |
| --- |
| 00 00 \* \* \* sh /backup/update\_neo4j/update.sh |

### 3.2.3 音频标签标注（1个程序）

1. **地址**

172.18.28.8

1. **代码及资源地址**

代码:

<http://172.21.151.6:82/svn/java/maven/quku/iMusicSong_PLF_1.0/scut/scut-MSDS/trunk/kg/src/scut_audioTagging/>

资源1:

<http://172.21.151.6:82/svn/java/maven/quku/iMusicSong_PLF_1.0/scut/scut-MSDS/trunk/kg/resource/>audio\_compile\_python369.tar

资源2:

<http://172.21.151.6:82/svn/java/maven/quku/iMusicSong_PLF_1.0/scut/scut-MSDS/trunk/kg/resource/>Miniconda3-latest-Linux-x86\_64.sh

1. **配置并调试程序**
2. 把svn路径[下编译完成的【资源1】audio\_compile\_python369.tar、conda](http://172.21.151.6:82/svn/java/maven/quku/iMusicSong_PLF_1.0/scut/scut-MSDS/trunk/下编译完成的audio_compile_python369.tar和conda)安装脚本【资源2】Miniconda3-latest-Linux-x86\_64.sh和【代码】复制到imusic\_4A用户目录下并切换到用户目录

cd /home/imusic\_4A

1. 解压编译好的python

tar -xvf audio\_compile\_python369.tar

1. 创建虚拟环境audio\_env

Python369/bin/python3 -m venv audio\_env

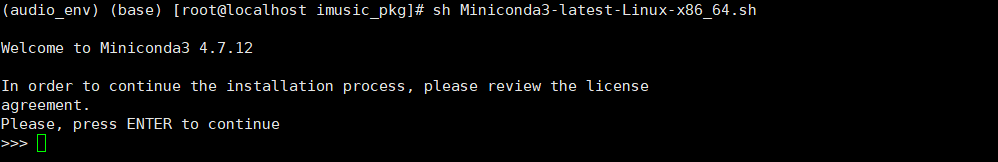
1. 激活虚拟环境audio\_env

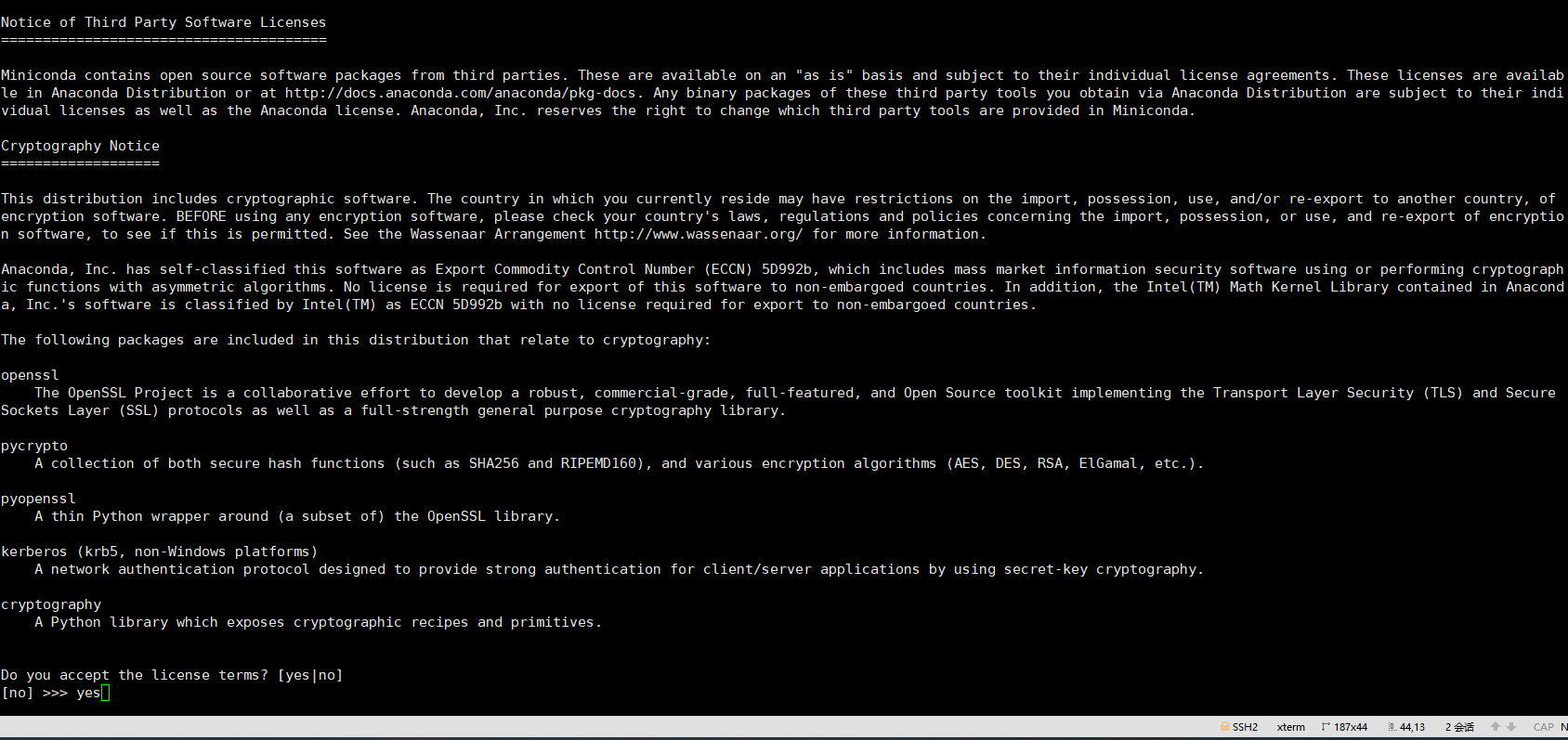
source /home/imusic\_4A/audio\_env/bin/activate

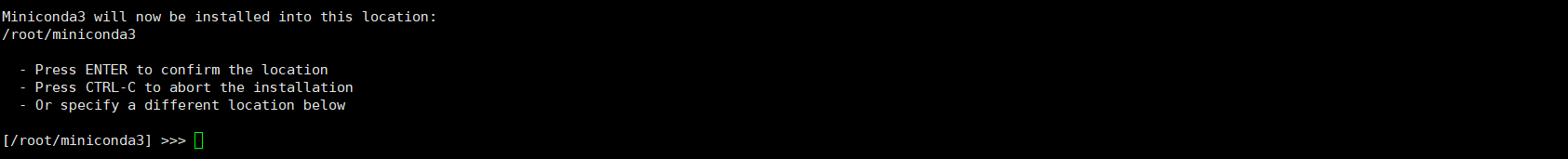
1. 安装miniconda

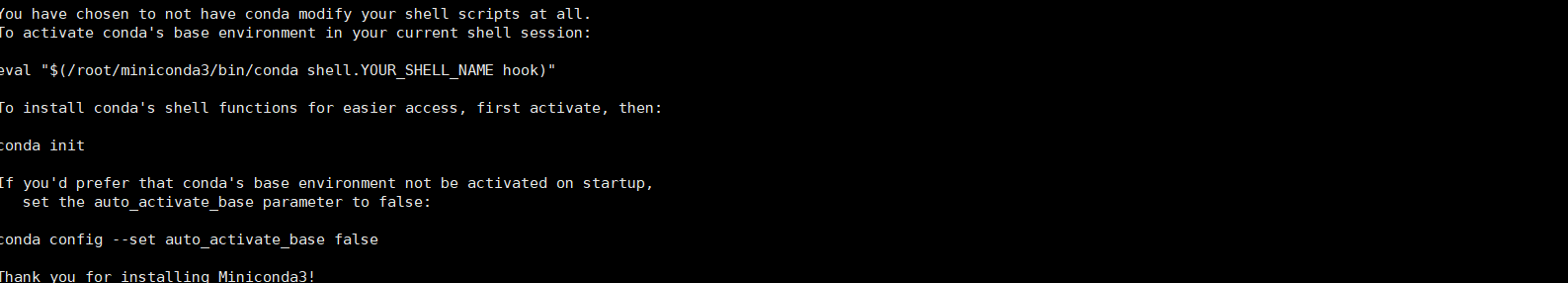
sh Miniconda3-latest-Linux-x86\_64.sh

# 一路按照默认配置进行

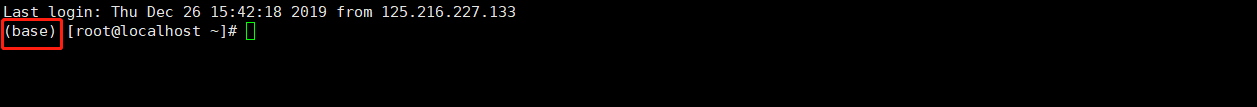








1. 重新登入，使conda环境生效



1. vi .condarc,修改为以下内容：

channels:

- https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free/

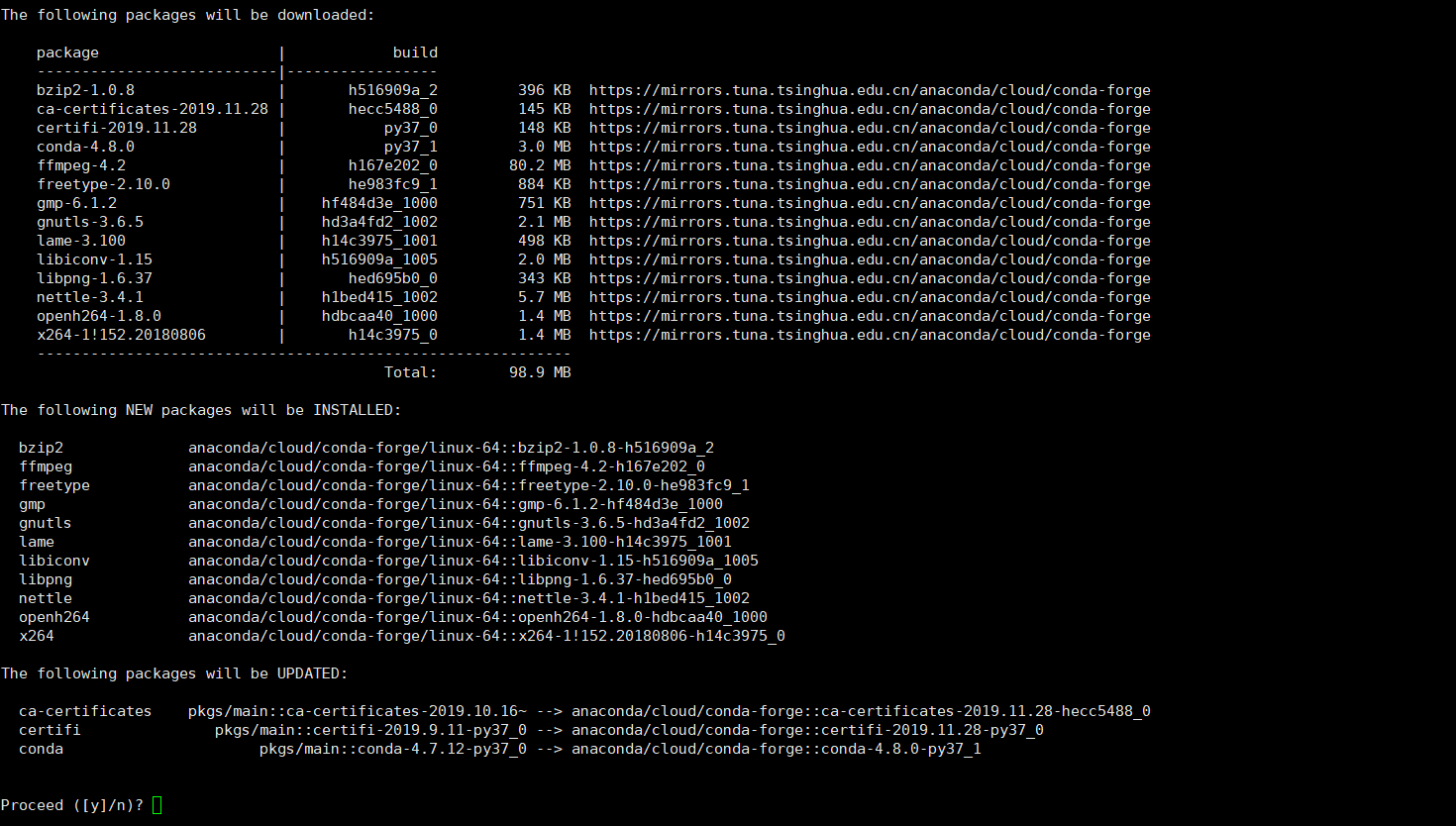
- https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/conda-forge/

- https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/msys2/

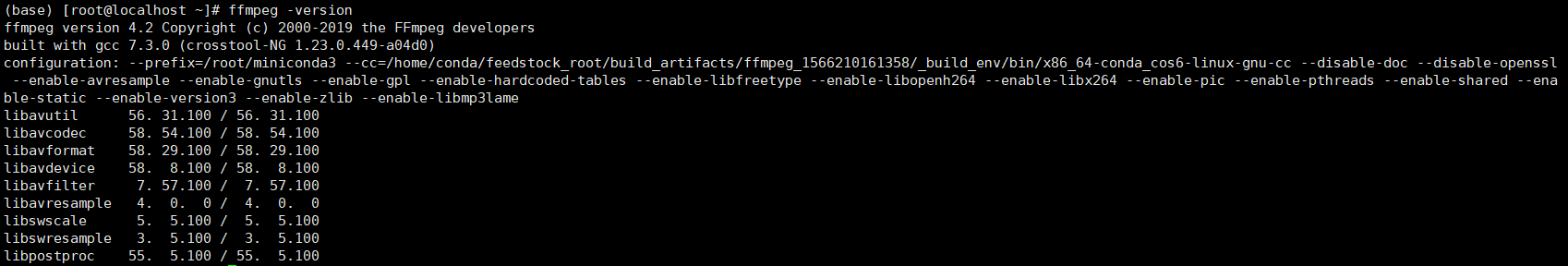
show\_channel\_urls: true

1. conda安装音频处理库ffmpeg

conda install ffmpeg



# 安装完成后，输入ffmpeg -version，可看到如下信息



1. 把svn中的代码复制到用户目录下并切换到代码目录

cd /backup/scut\_audioTagging

1. 下载必备的库

pip install -r requirements.txt

1. 配置数据库

cd musicTagging

vi scut\_audioTagging\_main.py

# 修改配置（57行左右）

# 查看host，user，password，port，db是否正确

# 可在poolSize控制进程池数量，根据机器内存大小和核心数而定，16G内存一般设为6个可稳定运行

|  |
| --- |
| 'poolSize': 6,  。。。  'dbConfig': {  'host': '172.17.104.16',  'user': 'read\_song',  'password': 'Ss#u3w#H8',  'port': 3307,  'db': 'musiclib',  'tag\_table': 'multimodal\_tag',  'song\_table':'search\_content',  'info\_column ': 'audio\_genre\_tag'} |



1. 调试程序

# 先杀掉上一个音频预测的程序

ps -ef | grep scut\_audioTagging\_main.py | grep -v grep| awk '{print $2}' | xargs kill -9

# 删掉缓存的mp3

rm -f \*.mp3

# 激活conda环境

source /home/imusic\_4A/miniconda3/bin/activate base

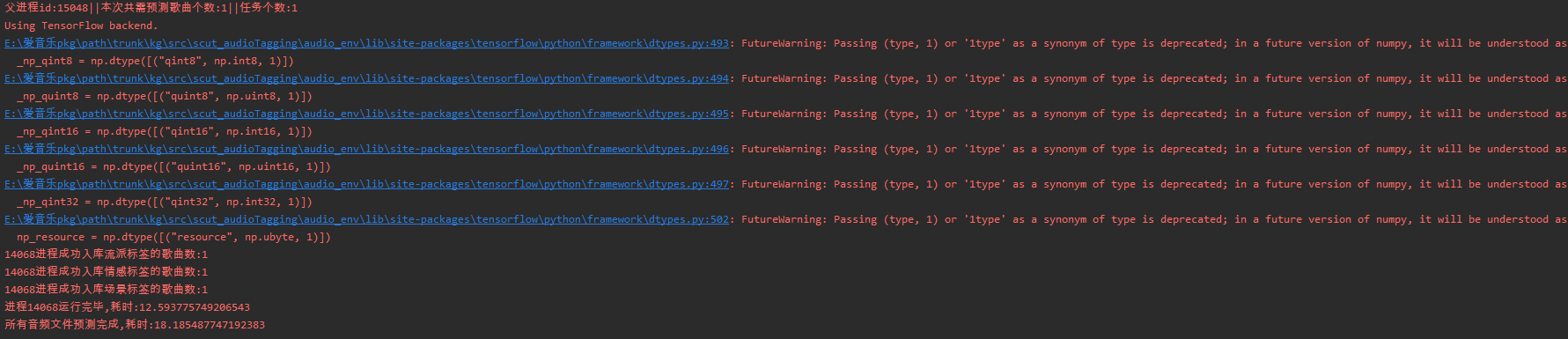
# 激活audio\_env虚拟环境

source /home/imusic\_4A/audio\_env/bin/activate

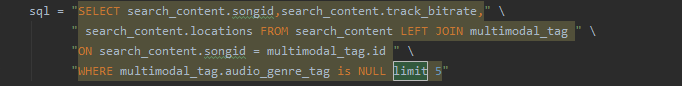
# 运行程序

python scut\_audioTagging\_main.py

# 如看到类似如下信息，则证明运行成功（忽略warning信息），因程序运行时间较久，可以先kill掉，为其设置定时任务；也可以在287行左右的sql语句上加上limit进行小规模调试

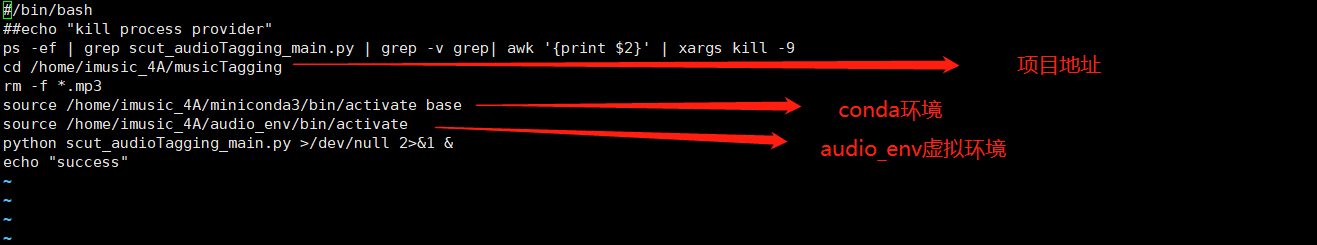


eg.以上为抽取5首歌的结果，只预测到1首歌是因为locations为none



1. 配置脚本，制定定时任务

**vi** run\_scut\_audio.sh



crontab -e

# 加入定时任务，每周日0点执行

|  |
| --- |
| 0 0 \* \* 0 sh /home/imusic\_4A/musicTagging/run\_scut\_audio.sh |

### 3.2.4 歌词标签标注+集成标签预测+人工标签修改（3个程序）

1. **地址**

172.18.28.7

* **代码及资源地址**

代码1：

<http://172.21.151.6:82/svn/java/maven/quku/iMusicSong_PLF_1.0/scut/scut-MSDS/trunk/kg/src/ensemble_tag/>

代码2：

<http://172.21.151.6:82/svn/java/maven/quku/iMusicSong_PLF_1.0/scut/scut-MSDS/trunk/kg/src/human_label/>

1. **安装python（yum安装）**
2. 切换root

yum install python36

1. 切换成普通用户imusic\_4A在/backup路径创建虚拟环境lrctag

python3 -m venv lrctag

1. **更改pip源，提高pip装库速度**
2. mkdir ~/.pip
3. touch ~/.pip/pip.conf
4. vi ~/.pip/pip.conf

# 更换成清华源，输入以下内容

|  |
| --- |
| [global]  index-url = <https://pypi.tuna.tsinghua.edu.cn/simple> |

1. **配置并调试歌词标注+集成预测的程序**
2. 复制代码到/backup目录并切换到代码目录

cd /backup/ensemble\_tag

1. 安装所需的python库

source /backup/kgbuild/bin/activate

pip install -r requirements.txt

1. 修改配置

vi settings.py

|  |
| --- |
| # 是否全量更新  FULL\_PREDICT = True  # 项目路径  # 注意此处末尾要加上斜杠  PROJECT\_PATH = '/backup/ensemble\_tag/'  # db  MYSQL\_DB\_HOST = '172.17.104.16'  MYSQL\_DB\_USER = 'read\_song'  MYSQL\_DB\_PASSWORD = 'Ss#u3w#H8'  MYSQL\_DB\_PORT = 3307  MYSQL\_DB = 'musiclib'  MYSQL\_SEARCH\_CONTENT = 'search\_content'  MYSQL\_MULTIMODAL\_TAG = 'multimodal\_tag'  #elasticsearch  ES\_CLIENT = 'esdata.imu.cn'  ES\_PORT = '8200'  ES\_USER = 'scutnetwork'  ES\_PASSWORD = 'scut7872=kaJIW'  ES\_BAIKE\_SONG = 'scut\_baike\_song'  ES\_BAIKE\_ACTOR = 'scut\_baike\_actor'  ES\_QQ\_SONG = 'scut\_qq\_song'  ES\_QQ\_SEARCH\_GEDAN = 'scut\_qq\_search\_gedan\_tag'  ES\_NETEASE\_SONG = 'scut\_netease\_song'  ES\_NETEASE\_SEARCH\_GEDAN ='scut\_netease\_search\_gedan\_tag' |





# 运行程序

# 激活虚拟环境

source /backup/lrctag/bin/activate

# 修改环境变量

export PYTHONPATH=/backup/ensemble\_tag:$PYTHONPATH

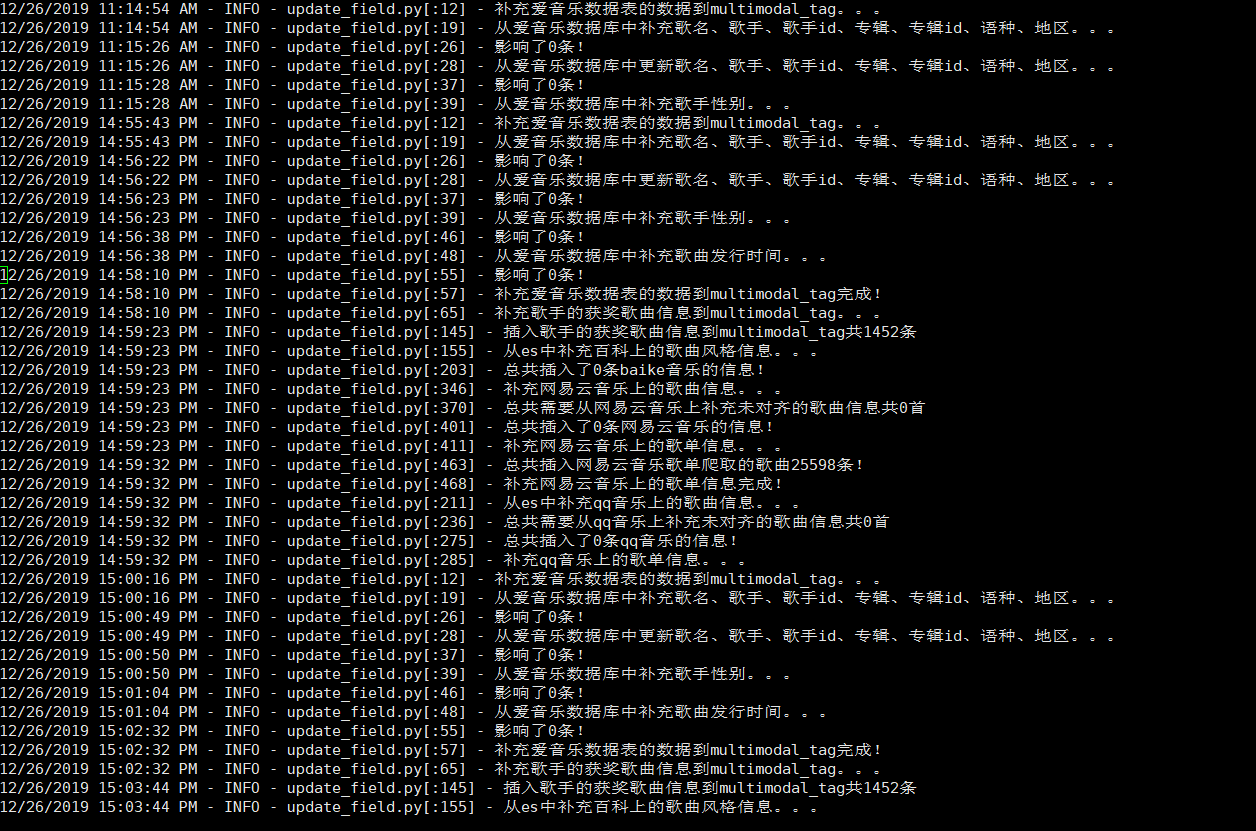
# 判断系统中是否已经有程序1【歌词标签标注+集成标签预测】在运行，如果有，请先kill掉

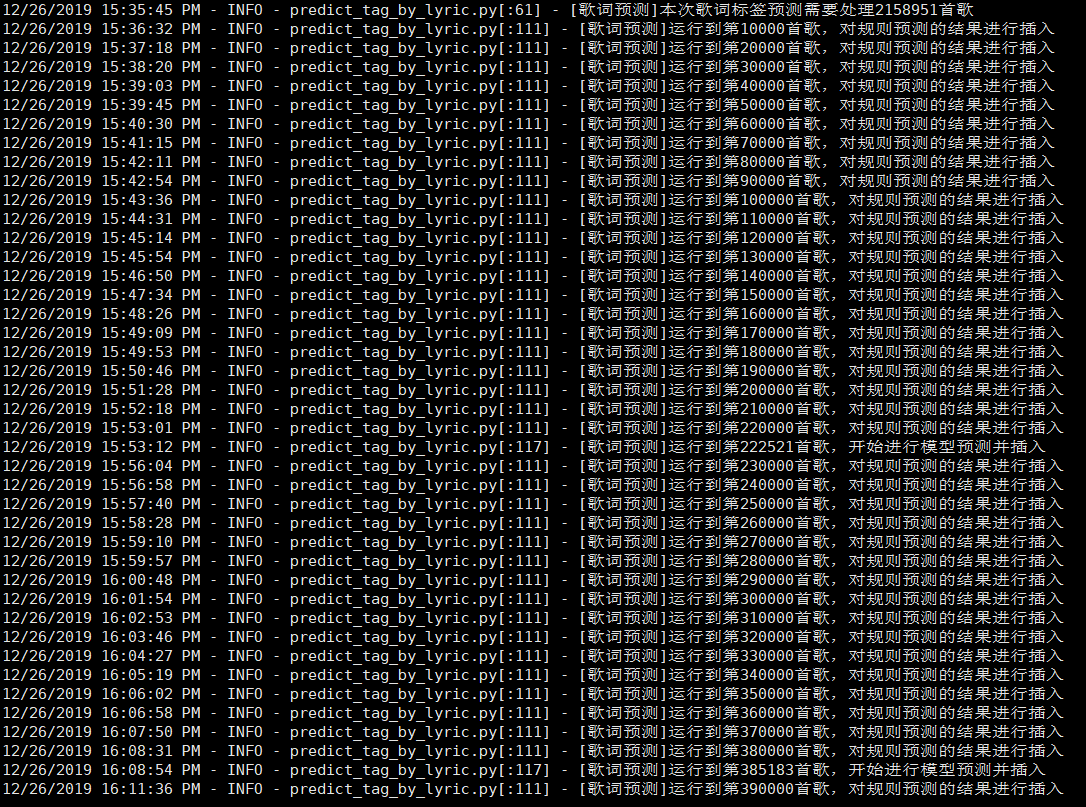
ps -ef|grep predict\_label\_procedure.py

# 运行程序1【歌词标签标注+集成标签预测】

python predict\_label\_procedure.py

# 程序运行时间可能比较久，因此看到如下信息并且无报错，则可以kill掉，为其设置定时任务

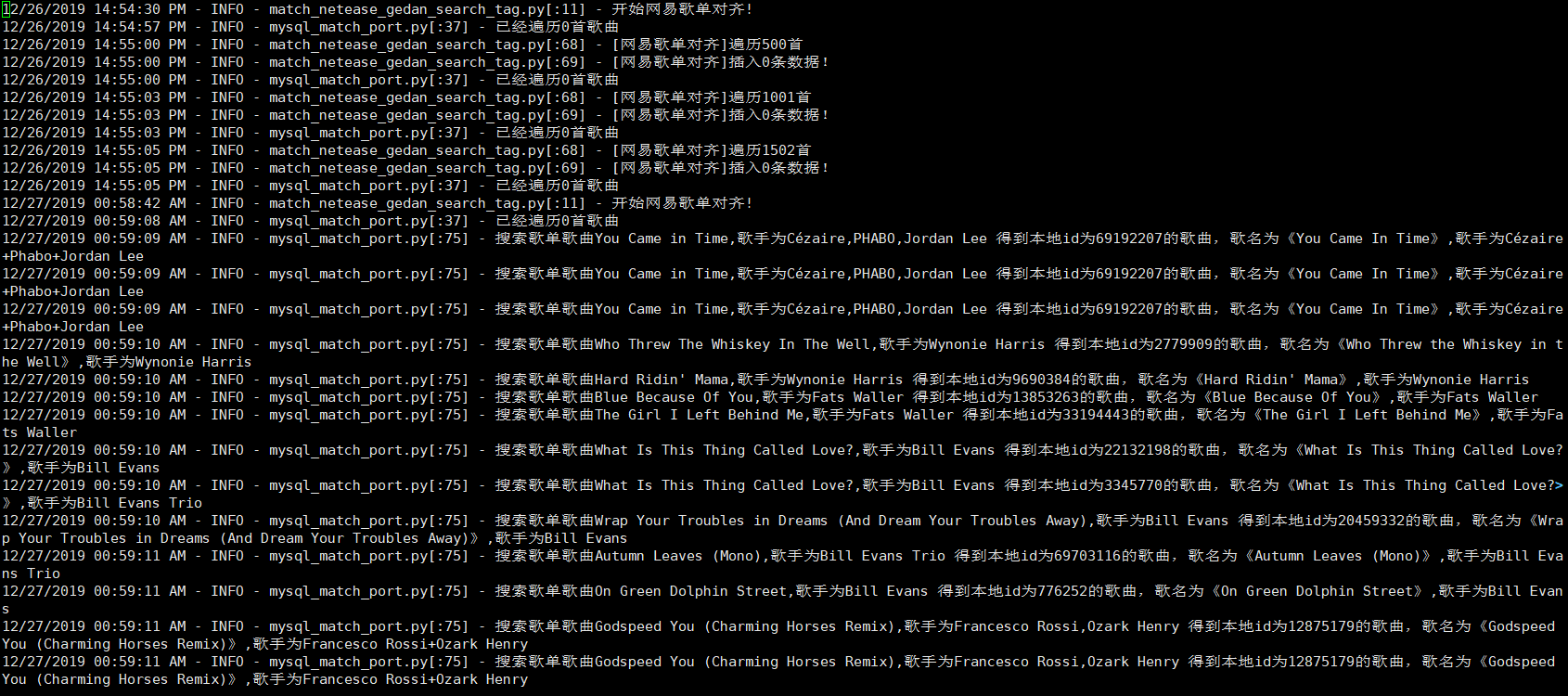




# 运行程序2【歌单标签对齐】

python gedan\_alignment/match\_gedan\_procedure.py

# 程序运行时间可能比较久，因此看到如下信息并且无报错，则可以kill掉，为其设置定时任务



1. **配置并调试人工标签修改的程序**

vi /backup/human\_label/settings.py

|  |
| --- |
| # app端口  APP\_PORT = 8080  # mysql配置  MYSQL\_HOST = '172.17.104.16'  MYSQL\_PORT = 3307  MYSQL\_USER = 'read\_song'  MYSQL\_PASSWORD = 'Ss#u3w#H8'  MYSQL\_DB = 'musiclib'  # neo4j地址  NEO4J\_URLS = ['http://172.18.28.4:7474','http://172.18.28.5:7474','http://172.18.28.6:7474']  # 更新solr地址  UPDATE\_SOLR\_URL = 'http://172.18.28.21:8080/putIndex' |

# 运行程序3【人工标签修改服务】

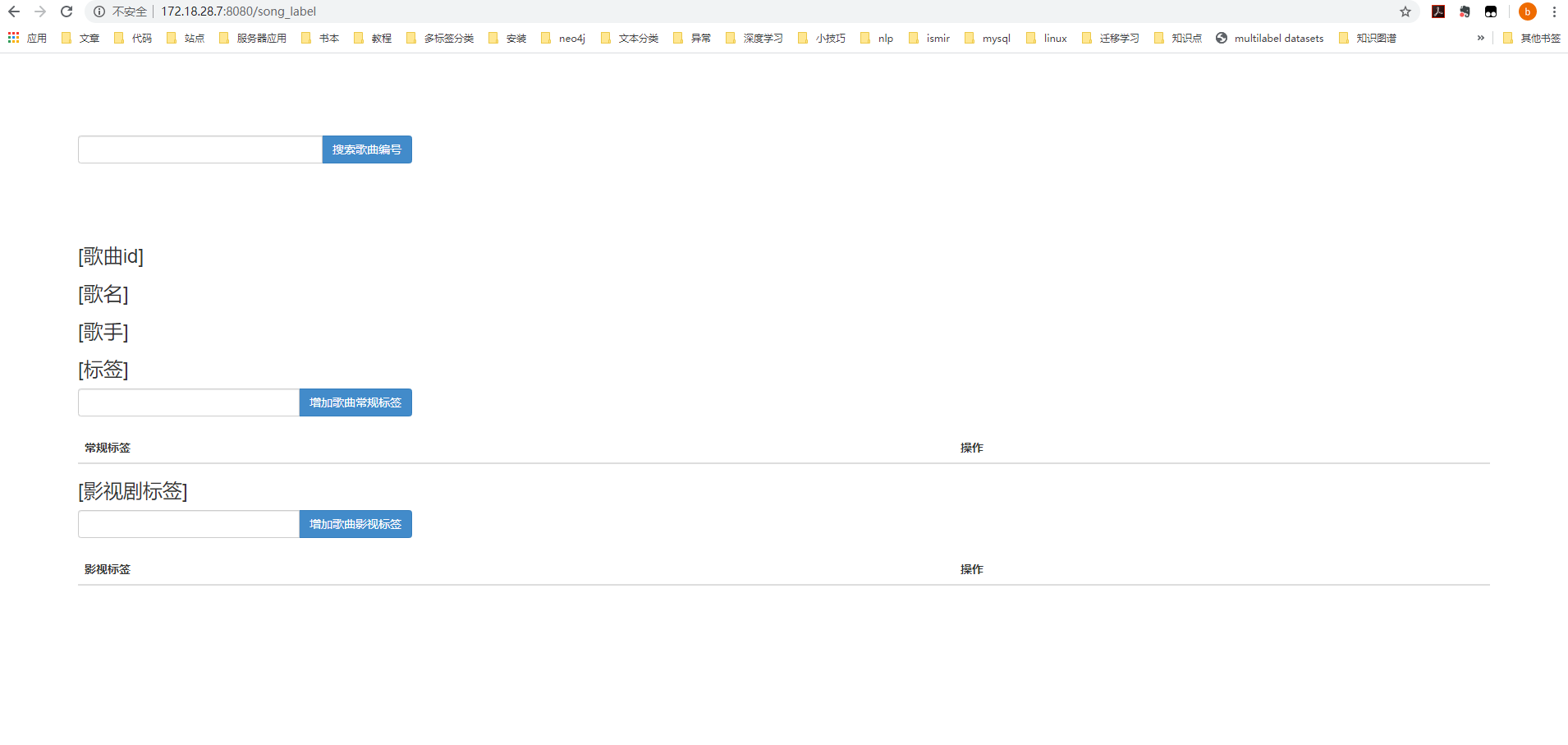
# 修改环境变量

export PYTHONPATH=/backup/human\_label:$PYTHONPATH

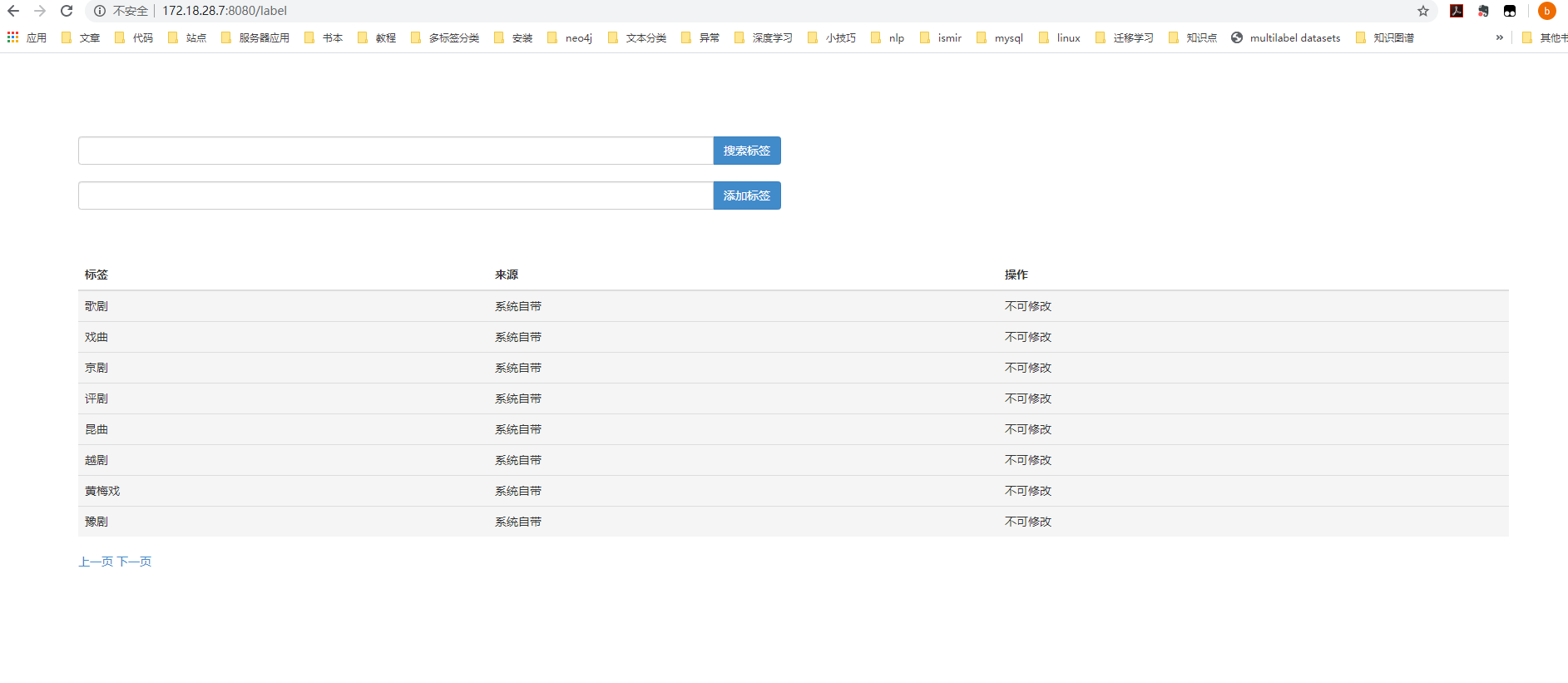
python /backup/human\_label/scut\_human\_label.py

# 打开下面两个链接，查看是否可以成功打开

<http://172.18.28.7:8080/song_label>

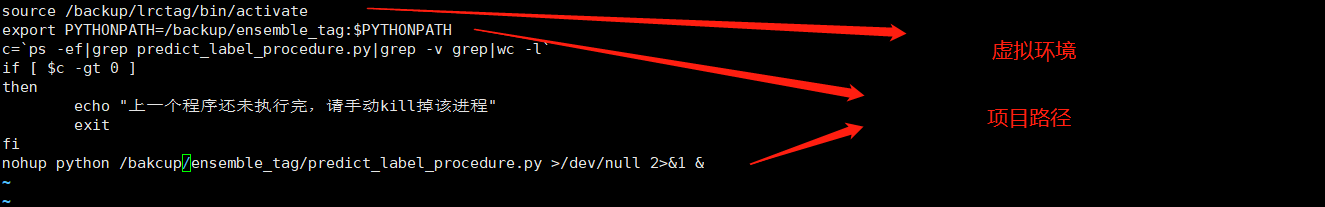


<http://172.18.28.7:8080/label>

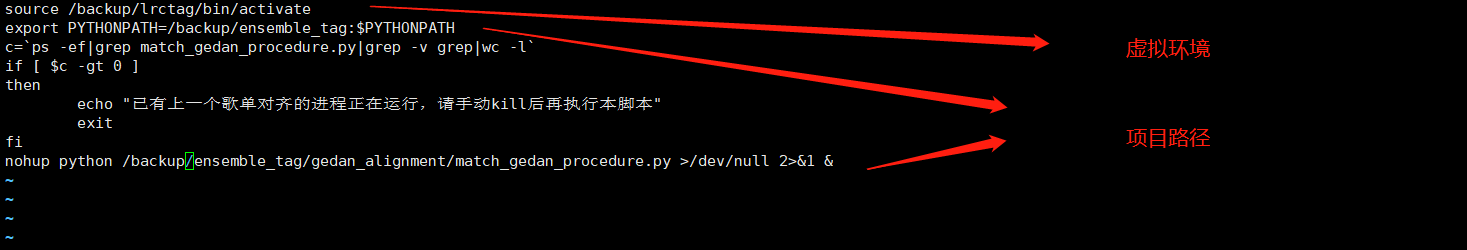


1. **配置脚本，制定定时任务**

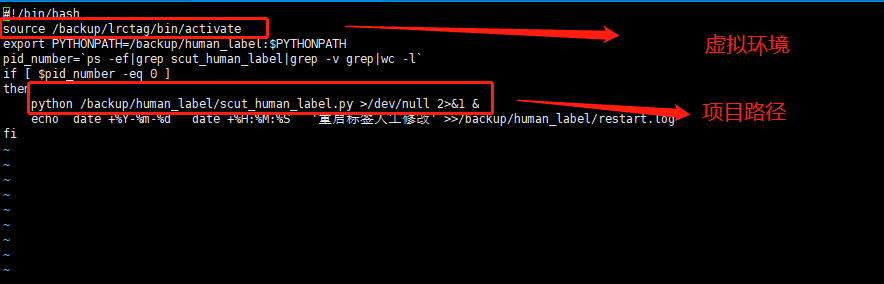
**vi** predict\_ensemble\_tag.sh

****

vi alignment\_gedan.sh

****

**vi** /backup/human\_label/restart\_human\_label.sh

****

crontab -e

# 加入定时任务，每天晚上11点开始运行歌词标签标注+集成标签预测程序

# 每周六0点开始运行歌单歌曲对齐程序

# 每分钟检查人工修改标签服务是否存活

|  |
| --- |
| 0 23 \* \* \* sh /backup/ensemble\_tag/predict\_ensemble\_tag.sh  00 00 \* \* 6 sh /backup/ensemble\_tag/alignment\_gedan.sh  \*/1 \* \* \* \* sh /backup/human\_label/restart\_human\_label.sh |

## 3.3 SolrCloud安装

下面安装步骤用 nodeX替代zookeepoer对应的IP，solrX替代solr对应的IP，solrPort替代solr对应的端口，参考此文档执行时需要注意。

### 3.3.1 zookeeper安装

1. 节点说明

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 用途 | ip | 账号 | CPU核数 | 内存 | 磁盘 |
| zookeeper | 172.18.254.52 | apptmc | 4 | 8G | 150G |
| zookeeper | 172.18.254.53 | apptmc | 4 | 8G | 150G |
| zookeeper | 172.18.254.54 | apptmc | 4 | 8G | 150G |
| zookeeper | 172.18.254.55 | apptmc | 4 | 8G | 150G |
| zookeeper | 172.18.254.56 | apptmc | 4 | 8G | 150G |

1. 安装步骤

以上列表为公司现有zookeeper集群，solrcloud直接基于此zookeeper集群构建。如需自行构建，参考如下步骤：

1. 下载安装包

<https://mirrors.tuna.tsinghua.edu.cn/apache/zookeeper/zookeeper-3.4.14/zookeeper-3.4.14.tar.gz>

1. 解压安装包

tar -zxvf zookeeper-3.4.14.tar.gz -C /backup/apps/

1. 修改zoo.cfg配置文件

cd /backup/apps/zookeeper-3.4.14/conf/

cp zoo\_sample.cfg zoo.cfg

vi zoo.cfg

修改存储快照文件snapshot的目录：

dataDir=/backup/apps/zookeeper-3.4.14/tmp

在最后添加（这里server.x中的x是一个数字，与myid文件中的id是一致的）：

server.1=zk1:8181:9181

server.2=zk2: 8181:9181

server.3=zk3: 8181:9181

server.4=zk4: 8181:9181

server.5=zk5: 8181:9181

保存退出

1. 创建一个tmp文件夹，存放myid

mkdir /backup/apps/zookeeper-3.4.14/tmp

echo 1 > /backup/apps/zookeeper-3.4.14/tmp/myid

1. 分发安装包

将配置好的zookeeper拷贝到其他节点，比如node2：

scp -r /backup/apps/zookeeper-3.4.14/ node2:/backup/apps/

注意：修改其余节点服务器对应/backup/apps/zookeeper-3.4.14/tmp/myid内容，比如，node2：

echo 2 > /backup/apps/zookeeper-3.4.14/tmp/myid

1. 启动Zookeeper集群（分别在每台服务器启动Zookeeper）

启动Zookeeper命令如下：

cd /backup/apps/zookeeper-3.4.14/bin/

./zkServer.sh start

查看状态：一个leader，其余follower，命令如下：

./zkServer.sh status

注意：一定要分别在所有节点执行启动命令来启动Zookeeper；上述步骤提到的zkX 替换为具体的IP。

### 3.3.2 solr安装及配置

1. 节点说明

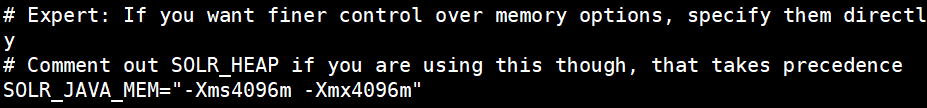
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 用途 | ip | 账号 | CPU核数 | 内存 | 磁盘 |
| solrcloud | 172.18.28.13 | imusic\_4A | 4 | 8G | 200G |
| solrcloud | 172.18.28.14 | imusic\_4A | 4 | 8G | 200G |
| solrcloud | 172.18.28.15 | imusic\_4A | 4 | 8G | 200G |
| solrcloud | 172.18.28.16 | imusic\_4A | 4 | 8G | 200G |
| solrcloud | 172.18.28.17 | imusic\_4A | 4 | 8G | 200G |
| solrcloud | 172.18.28.18 | imusic\_4A | 4 | 8G | 200G |
| solrcloud | 172.18.28.19 | imusic\_4A | 4 | 8G | 200G |
| solrcloud | 172.18.28.20 | imusic\_4A | 4 | 8G | 200G |

1. 配置cloud模式

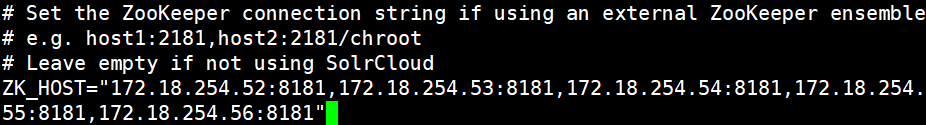
solr5.0之后官方建议使用自带的jetty代替tomcat容器。使用jetty更容易部署cloud模式。

解压solr-7.7.1.tgz，进入bin目录，修改solr.in.sh文件：

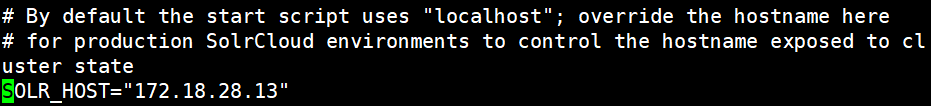
1. 修改jvm内存大小



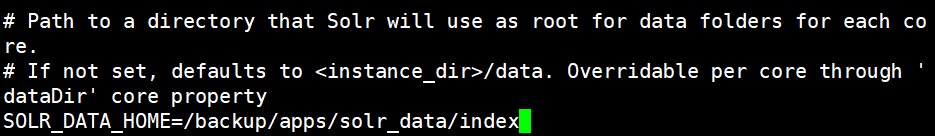
1. 设置ZK\_HOST，一定要设置这个参数，否则solr会以单机运行



1. 设置本机host



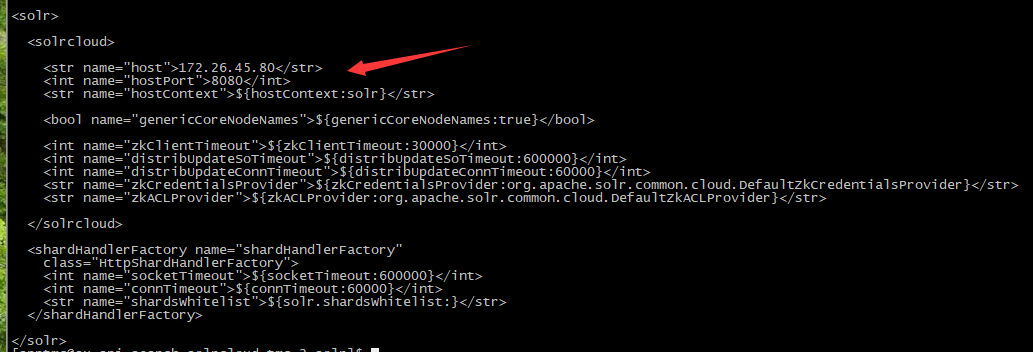
1. 设置索引数据存放的目录



1. jetty运行端口



修改solr-7.7.1/server/solr目录下的solr.xml文件，改成服务器对应的ip



* 启动与停止

1. 启动solr-7.7.1/bin/solr start
2. 停止solr-7.7.1/bin/solr stop

* Solr自定义分词

测试发现，公司原有部署使用的jcseg2.4.0在使用词库lexicon的模式，比如complex时出现错误，通过测试并与开发者联系确定以下版本正常运行。

|  |  |
| --- | --- |
| Jcseg版本 | Solr版本 |
| 1.9.6 | 5.2.1 |
| 2.5.0 | 7.7.1， 8.2.0 |

1. 下载2.5.0的分词包源码，并mvn clean package编译,得到：

jcseg-analyzer-2.5.0.jar

jcseg-core-2.5.0.jar

<http://172.21.151.6:82/svn/java/maven/quku/iMusicSong_PLF_1.0/scut/scut-MSDS/trunk/solr/jcseg-2.5.0>

svn上同样保存了编译好的两个jar包，可直接使用。

1. 替换原有的2.4.0

在 server/solr-webapp/webapp/WEB-INF/lib 目录下，删除原有的

jcseg-analyzer-2.4.0.jar

jcseg-core-2.4.0.jar

并复制上述的2.5.0的两个jar包。

1. 爱音乐自定义分词

com-imusic-comma-tokenizer.jar

com-imusic-query-parser.jar

<http://172.21.151.6:82/svn/java/maven/quku/iMusicSong_PLF_1.0/scut/scut-MSDS/trunk/solr>

svn上有保存，可直接使用。

1. 重新启动solrcloud

每个节点上的solr都重新启动：./bin/solr restart

### 3.3.3 添加collection

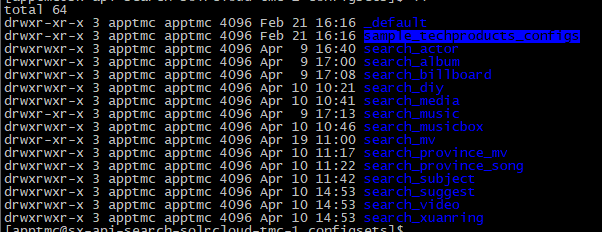
* 配置文件地址

<http://172.21.151.6:82/svn/java/maven/quku/iMusicSong_PLF_1.0/scut/scut-MSDS/trunk/solr/conf/msds_music>

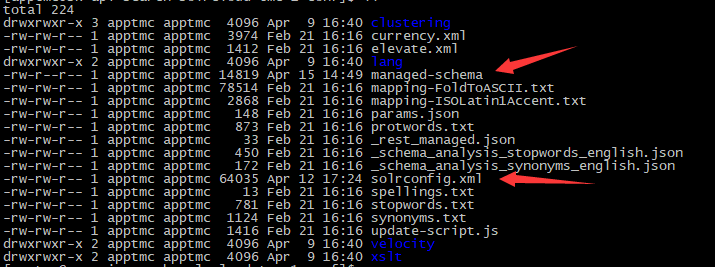
msds\_music 这个collection为搜索和对话系统依赖，配置文件在svn，因此下一步操作中的“配置文件准备”步骤可以跳过。

* 添加msds\_music collection

1. 配置文件准备
2. 在solrcloud模式下配置文件存放在zookeeper上。在server/solr/configsets目录下复制一份sample\_techproducts\_configs配置文件。



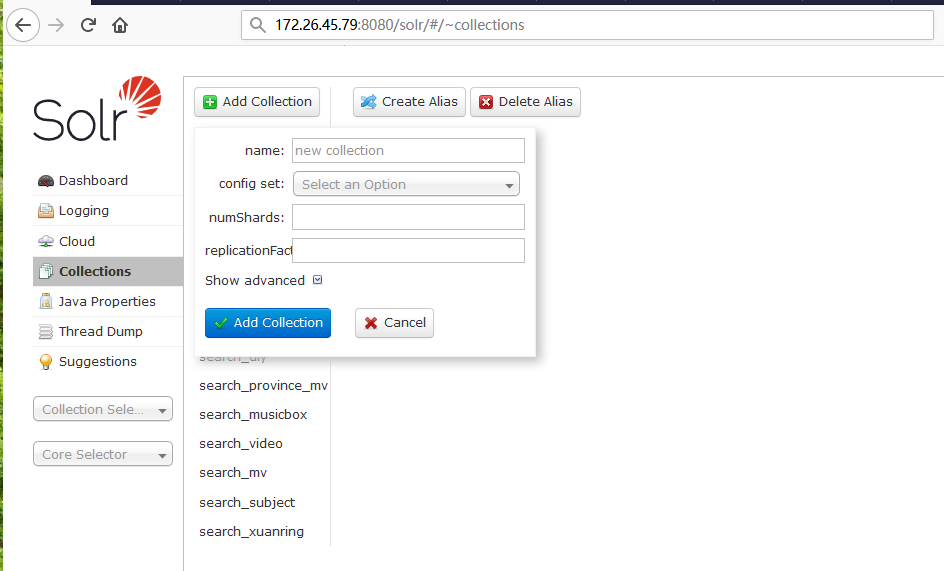
1. 进入目录下按照需求修改managed-schema（相对于schema.xml文件）和solrconfig.xml文件。



1. 文件上传到zookeeper

进入目录solr-7.7.1/server/scripts/cloud-scripts下执行zkcli.sh，把配置文件上传到zookeeper，如：./zkcli.sh -zkhost zk1:zkPort,zk2: zkPort, zk3: zkPort,zk4: zkPort,zk5: zkPort -cmd upconfig -confdir /backup/bak/solr7\_ext/conf/msds\_music/ -confname msds\_music

1. 添加collection
2. 上传完成后，到solr管理界面http://solr1:solrPort/solr/#/~collections（可任意一个服务节点）添加collection。



1. 填写collection name，config set选择刚才上传的配置，填写分片数（numShards）和每个分片的副本数（replicationFactor）。
2. 在cloud的tab中可以看到每个collection的健康状态。
3. 另一种方式创建collection

#!/bin/bash

SOLR\_BASE=/backup/apps/solr-7.7.1

COLLECTION=msds\_music

PORT=8080

$SOLR\_BASE/bin/solr create -c $COLLECTION -d ./conf/$COLLECTION -shards 3 -replicationFactor 2 -p $PORT

在solr admin页面创建有时会超时，可以通过以上脚本进行collection的创建。

1. 修改配置文件

如果要修改索引的配置文件，则修改完配置文件信息之后，重新上传到zookeeper，然后在浏览器执行下面请求，reload信息。

<http://solr1:solrPort/solr/admin/collections?action=RELOAD&name=msds_music>

action：请求动作

name：collection name

* 验证服务

节点的页面没有提示错误，server/logs/solr.log中没有错误输出表示正常。

* 设置进程监控

crontab –e

\* \* \* \* \* sh /backup/apps/solr7\_ext/monitor\_solr.sh

## 3.4 索引更新子系统安装

* 下载最新版本索引更新子系统代码

<http://172.21.151.6:82/svn/java/maven/quku/iMusicSong_PLF_1.0/scut/scut-MSDS/trunk/solr/msds-solr7-importer>

到/backup/apps目录

* 编译

mvn clean package

* 将jar包从下面位置复制到需要的位置

msds-solr7-importer\target\msds-solr7-importer-0.1.0.jar

* 从下面位置复制配置文件到jar包同目录并修改

msds-solr7-importer\src\main\resources\application.properties

* 复制sh文件，并检查内部配置是否合理

monitor\_msds\_solr.sh

* 设置进程状态监控

crontab –e

\* \* \* \* \* sh /backup/msds\_solr\_importer/monitor\_msds\_solr.sh

* 周天到周五每天一次安全更新：

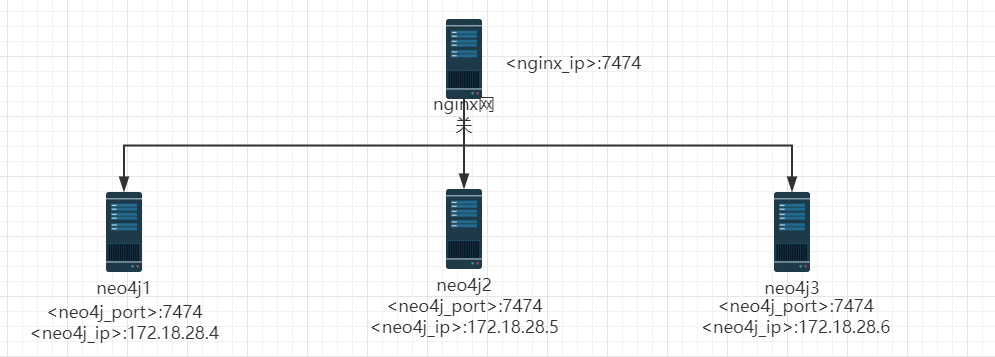
0 1 \* \* 0-5 sh /backup/apps/solr\_importer/update\_solr.sh update

* 每周六一次重置更新：

0 1 \* \* 6 sh /backup/apps/solr\_importer/update\_solr.sh reset

# 集群设计

## 4.1 neo4j集群



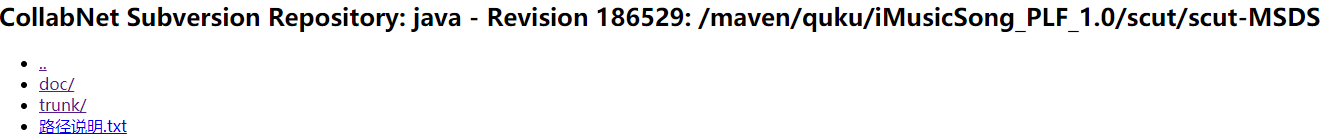
# 对话系统部署

## 5.1 部署路径

1) 搜索程序安装路径

/backup/apps

2) 代码和文档路径



<http://172.21.151.6:82/svn/java/maven/quku/iMusicSong_PLF_1.0/scut/scut-MSDS/>

① 文档位置

doc/

② 代码位置

|  |
| --- |
| trunk/  dialog/ //对话系统代码  resource/ //资源、模型  src/ // 源代码  kg/ //知识图谱代码  solr/ //solr代码  search/ //搜索代码 |

## 5.2 程序安装

* 安装Python环境

1. 下载Python源码<https://www.python.org/ftp/python/3.6.8/Python-3.6.8.tgz> 并解压；
2. 系统依赖安装

yum install openssl-devel

yum install zlib-devel

yum install bzip2-devel

yum install xz-devel

1. 配置，安装

./configure –prefix=/home/imusic\_4A/${pythonDir}

make

make install

* 安装依赖

1. Pip 国内镜像

vi ~/.pip/pip.conf 写入如下内容.

[global]

index-url = https://pypi.tuna.tsinghua.edu.cn/simple

1. 创建虚拟环境

${pythonDir}/bin/python –m venv iss\_env

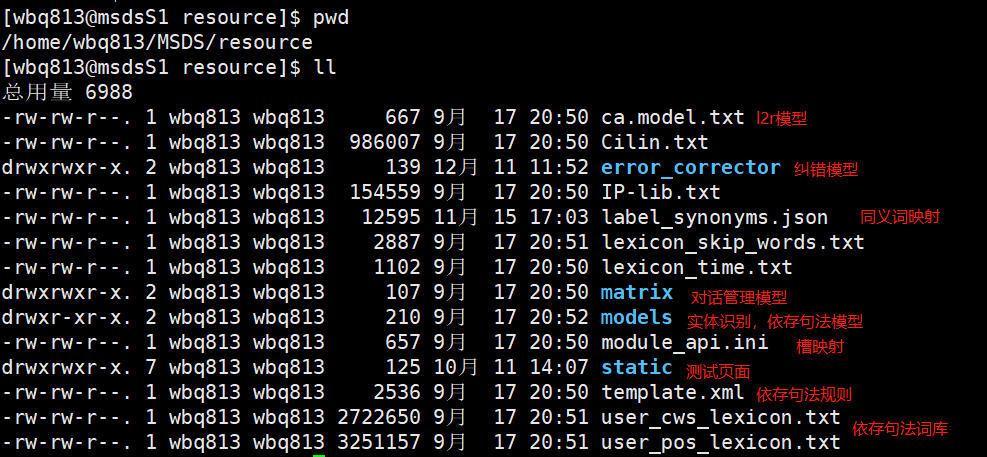
1. 安装依赖

source msds\_env/bin/activate

pip install –r MSDS/requirement.pip

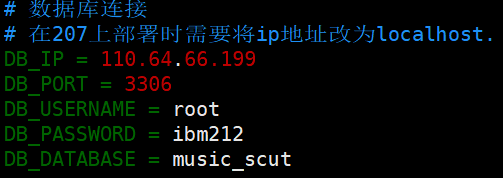
* 放入模型及资源文件

将资源压缩包解压到系统目录下的resource目录



* 修改配置文件

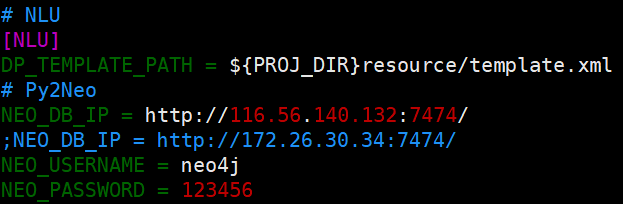
修改曲库连接信息：



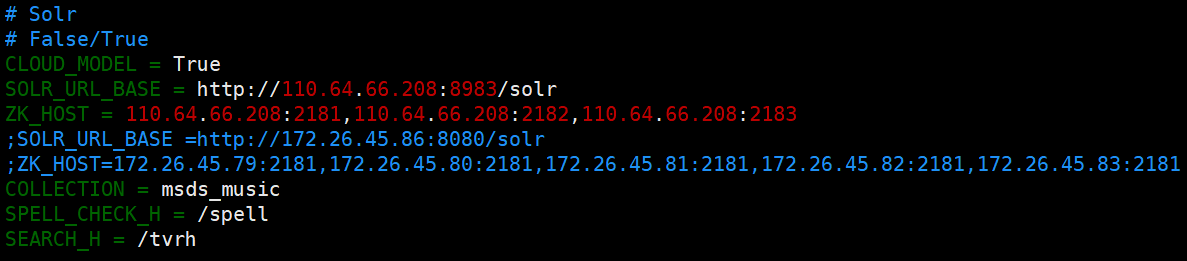
修改Redis连接信息：



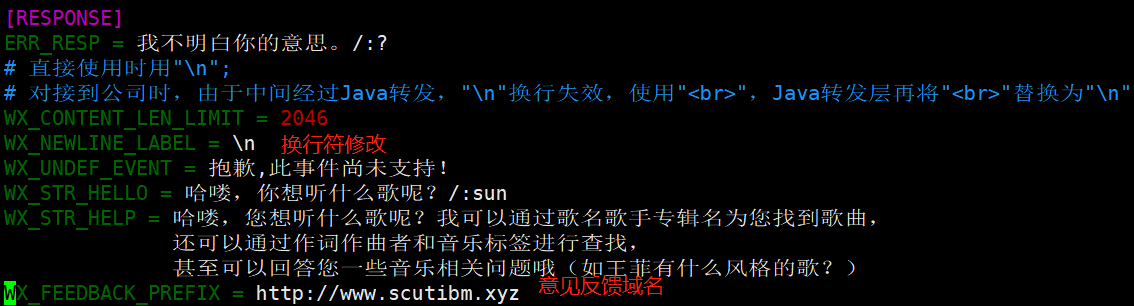
修改知识图谱连接信息：



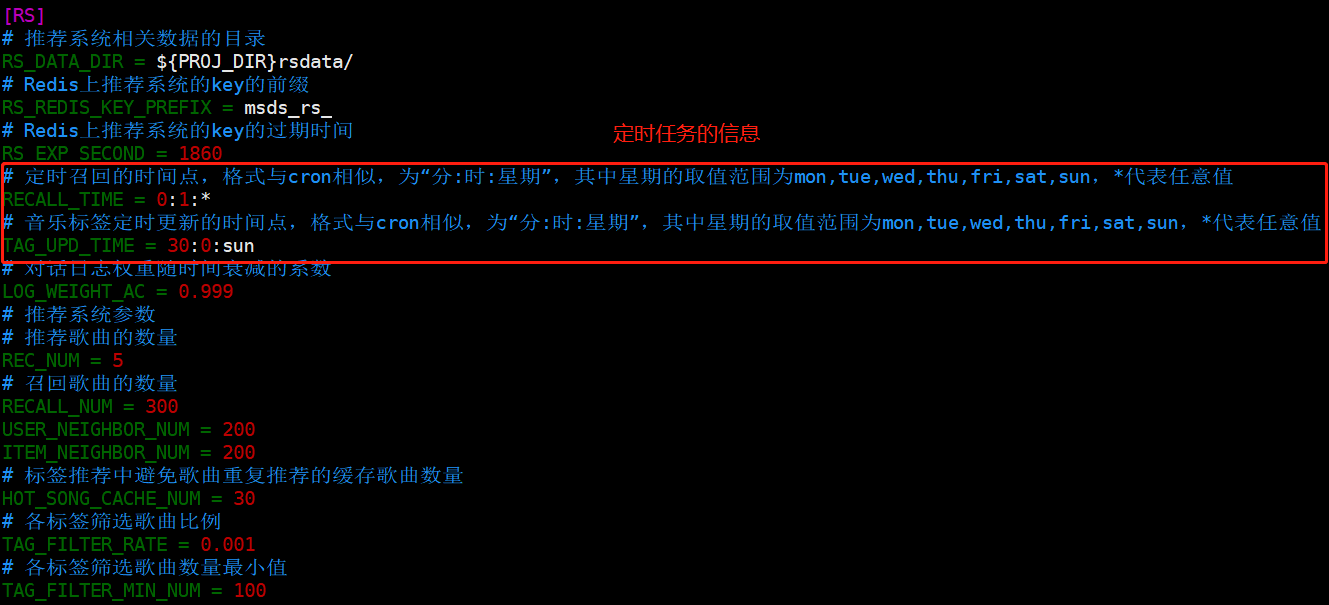
修改solr连接信息：



修改微信响应相关配置：这里特别注意换行符，在公司服务器运行需要修改为“<br>”.还有对应的反馈公网域名。

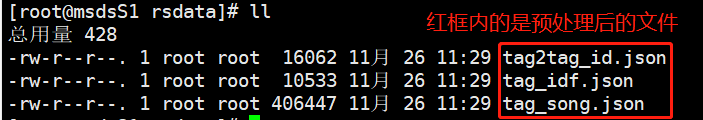


修改推荐相关配置：确保定时任务的信息与后面使用crontab设置定时程序的信息相对应，例如crontab标签定时更新任务设在星期天0时0分执行，则设置TAG\_UPD\_TIME = 30:0:sun，即程序的定时任务应比crontab的定时任务的执行时间晚30分钟。值得注意的是，定时召回任务的时间最好比标签定时更新任务的晚30分钟，避免同时运行给服务器带来较重的负荷。



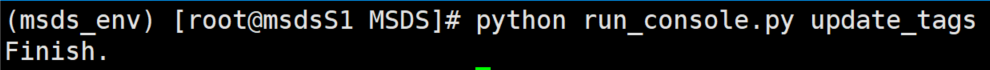
* 数据预处理

如果相应路径下没有预处理后的文件，则需要进行数据预处理



先source msds\_env/bin/activate 激活python虚拟环境

然后输入命令：python run\_console.py update\_tags



* 启动程序

启动程序前需确保有预处理后的文件

先source msds\_env/bin/activate 激活python虚拟环境

console模式：python run\_console.py

Web模式：./msds.sh start

* 启动定时程序

crontab -e编辑crontab文件，添加下面的定时任务

30 0 \* \* \* bash /backup/apps/{path}/update\_data.sh recall

0 0 \* \* 0 bash /backup/apps/{path}/update\_data.sh tag\_update

其中bash /backup/apps/{path}/update\_data.sh recall是召回任务

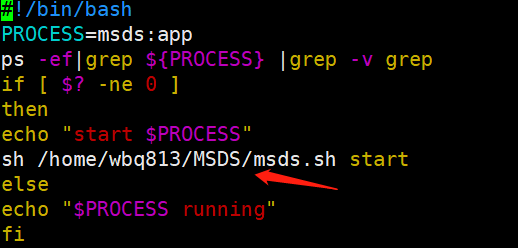
bash /backup/apps/{path}/update\_data.sh tag\_update是标签更新任务

eg: 30 0 \* \* \* bash /home/imusic\_4A/MSDS-v1.0/update\_data.sh recall

0 0 \* \* 0 bash /home/imusic\_4A/MSDS-v1.0/update\_data.sh tag\_update

* 启动监控程序

1. 修改monitor\_msds.sh脚本



2．crontab -e编辑crontab文件，添加下面的定时任务

\* \* \* \* \* sh /backup/apps/{path}/monitor\_msds.sh

eg：\* \* \* \* \* sh /home/imusic\_4A/MSDS-v1.0/monitor\_msds.sh

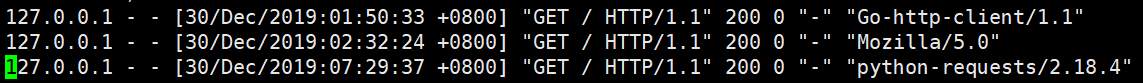
## 5.3 验证服务

Console模式下，可以直接输入语句测试，Web模式下需要在浏览器访问测试（访问地址为<http://172.26.13.16:8080/>）。

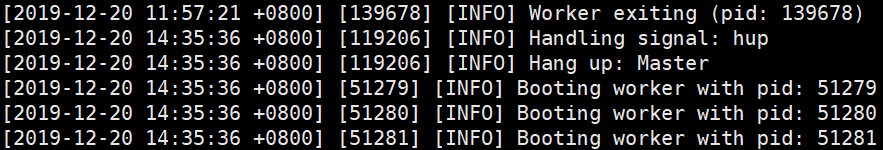
## 5.4 日志文件

日志文件保存在MSDS/logs/

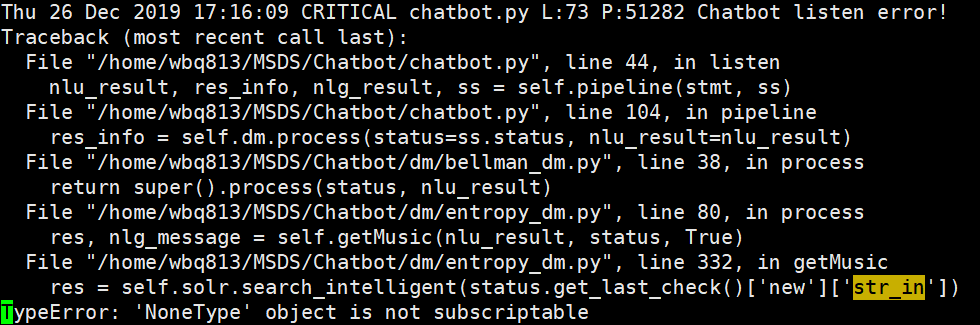
msds\_access.log 访问日志



msds\_gunicorn.log gunicorn服务器的日志



msds.log.2019-12-06 异常级别以上的日志



## 5.5 端口信息

Solr默认采用8080

Neo4j默认采用 8080

对话系统默认采用8080

## 5.6 程序停止

./msds.sh stop

# QA

## 部署问题及解决办法

1. Process finished with exit code 139 (interrupted by signal 11: SIGSEGV)

Chatbot/util/l2r中分词使用基于C语言的pyseg，使用conda安装的python环境会报错。

1. 访问二期系统的链接：

内网：

[http://scsearch.imu.cn/imusic/query.do?cmd=QueryMusic4Client&Keyword=%s&PageIndex=%s&PageSize=%s&Channel=CLIENT](http://scsearch.imu.cn/imusic/query.do?cmd=QueryMusic4Client&Keyword=%25s&PageIndex=%25s&PageSize=%25s&Channel=CLIENT)

外网（白名单）：

[http://scsearch.oe.imusic.cn/imusic/query.do?cmd=QueryMusic4Client&Keyword=%s&PageIndex=%s&PageSize=%s&Channel=CLIENT](http://scsearch.oe.imusic.cn/imusic/query.do?cmd=QueryMusic4Client&Keyword=%25s&PageIndex=%25s&PageSize=%25s&Channel=CLIENT)

1. jcseg与solr版本不合适的问题

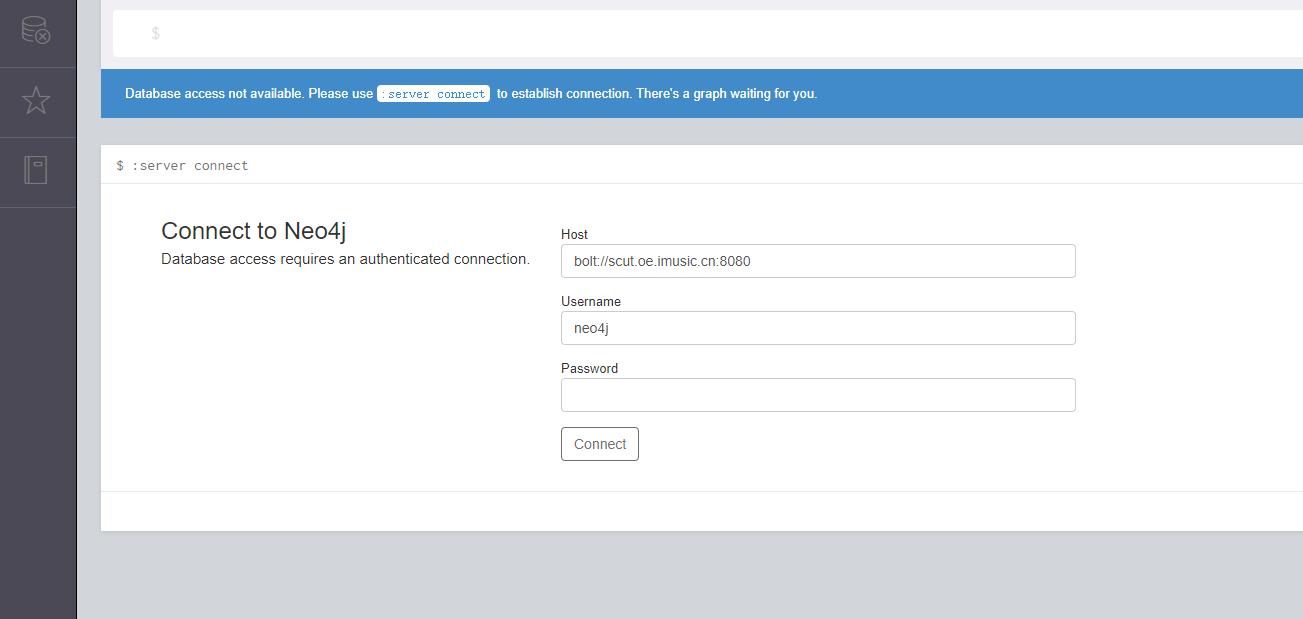
jcseg 2.4.0,2.6.0（未release）在solr7.7.1，8.2.0 下使用lexicon的时候均出现以下错误：

2019-10-30 01:08:19.530 ERROR (qtp898557489-31) [c:msds\_music s:shard1 r:core\_node5 x:msds\_music\_shard1\_replica\_n2] o.a.s.h.RequestHandlerBase org.apache.solr.common.SolrException: Exception writing document id 1938012 to the index; possible analysis error: startOffset must be non-negative, and endOffset must be >= startOffset, and offsets must not go backwards startOffset=412,endOffset=413,lastStartOffset=1746 for field 'originalrc'

经过测试和与开发者联系，经过测试,以下组合正常运行：

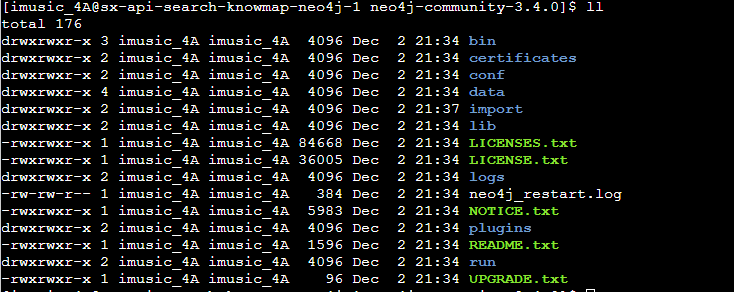
|  |  |
| --- | --- |
| Jcseg版本 | Solr版本 |
| 1.9.6 | 5.2.1 |
| 2.5.0 | 7.7.1、 8.2.0 |

1. 为什么访问直接访问<http://scut.oe.imusic.cn:80>会出现以下页面



答:因为通过浏览器访问neo4j需要bolt协议进行身份验证，而bolt协议的端口是8080,nginx代理只对neo4j集群的7474端口用80端口进行代理，并没有代理8080接口，所以直接访问<http://scut.oe.imusic.cn:80>不会像访问单台neo4j服务器那样可以在上方的命令行界面进行各种操作。

1. neo4j目录介绍



bin目录

启动脚本

data

数据存放地址，关闭neo4j后删除其中的database/graph.db则为清空数据库

plugins

apoc插件存放地址

1. ./iss.sh start 启动系统后无法访问。由于系统启动时需要加载一些后序处理用到的文件，这个过程耗时较长，导致超时，可以在gunicorn.conf中增大timeout参数来解决该问题。
2. 对话系统启动失败。可能是虚拟环境的依赖没有更新所导致的。

source msds\_env/bin/activate

pip install –r MSDS/requirement.pip

如果依赖已更新，但系统仍启动失败，则可能是因为gunicorn的worker数量过多， 服务器承受不了，可以修改gunicorn配置，减少worker数量。

