官方文档

https://lucene.apache.org/solr/guide/7\_4/deployment-and-operations.html

# 开始安装

## 、安装jdk8

yum install java

## 、配置limit

修改/etc/security/limits.conf文件

\* soft nofile 65535

\* hard nofile 65535

\* soft nproc 65535

\* hard nproc 65535

重启服务器

## 、下载并安装solr

wget http://mirrors.shuosc.org/apache/lucene/solr/7.6.0/solr-7.6.0.tgz

tar -zxvf solr-7.6.0.tgz

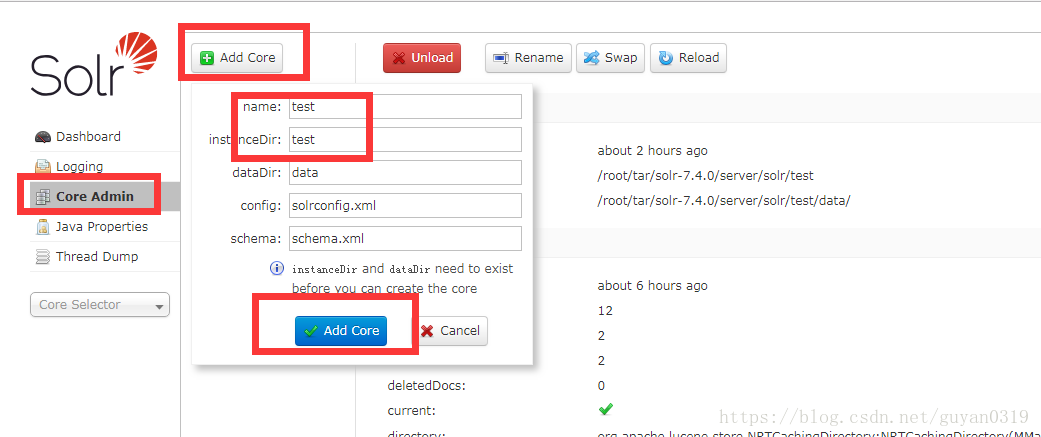
## 、复制新项目的配置文件

cd /root/tar/solr-7.6.0

mkdir server/solr/test

cp /usr/local/solr/solr-7.6.0/server/solr/configsets/\_default/conf/ /usr/local/solr/solr-7.6.0/server/solr/test -r

## 、访问 <http://192.168.153.136:8983> 并创建项目test



# 配置ikanalyzer和自带的中文分词器

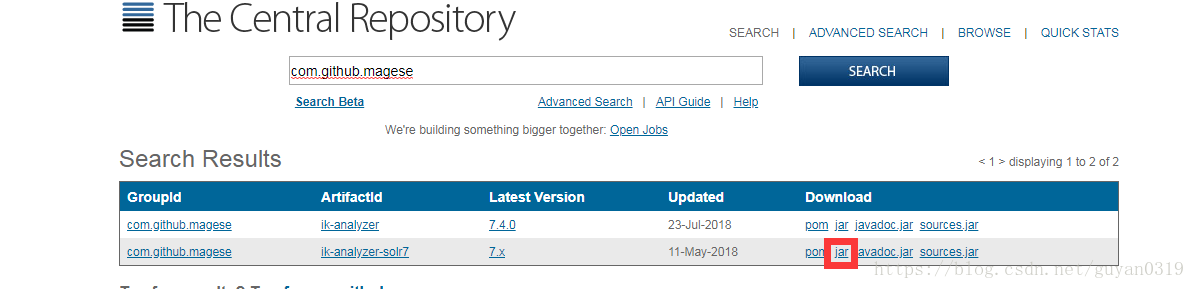
一般解决分词问题会选择ikanalyzer，因为相对来说ikanalyzer更新的比较的好， solr7 本身提供中文的分词jar包，在此一并讲一下。

## 使用ikanalyzer分词

### 先下载solr7版本的ik分词器，

下载地址：http://search.maven.org/#search%7Cga%7C1%7Ccom.github.magese

分词器GitHub源码地址：https://github.com/magese/ik-analyzer-solr7



将下载好的jar包放入solr-7.4.0/server/solr-webapp/webapp/WEB-INF/lib目录中

### 、复制新项目的配置文件

[root@localhost conf]# cd /usr/local/solr/solr-7.6.0/server/solr

[root@localhost conf]# mkdir ik

[root@localhost conf]# cp /usr/local/solr/solr-7.6.0/server/solr/configsets/\_default/conf/ /usr/local/solr/solr-7.6.0/server/solr/ik -r

### 、 然后到server/solr/ik/conf目录中打开managed-schema文件，增加如下代码

<!-- ik分词器 -->

<fieldType name="text\_ik" class="solr.TextField">

<analyzer type="index">

<tokenizer class="org.wltea.analyzer.lucene.IKTokenizerFactory" useSmart="false" conf="ik.conf"/>

<filter class="solr.LowerCaseFilterFactory"/>

</analyzer>

<analyzer type="query">

<tokenizer class="org.wltea.analyzer.lucene.IKTokenizerFactory" useSmart="true" conf="ik.conf"/>

<filter class="solr.LowerCaseFilterFactory"/>

</analyzer>

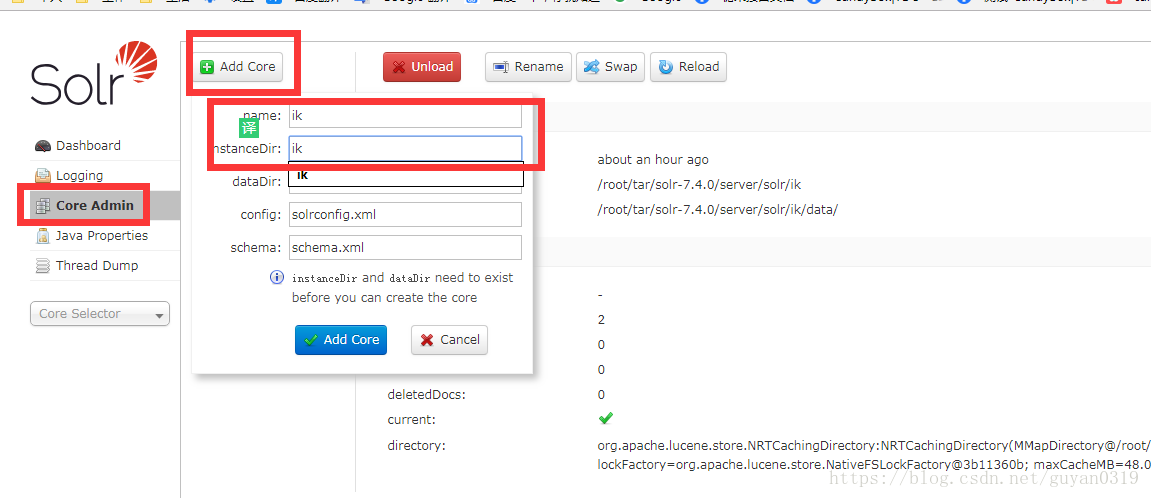
</fieldType>

### 、重启solr

bin/solr restart -force

### 、新增项目ik

访问项目 http://192.168.153.136:8983



## solr自带分词中文分词器

### 、复制jar包

cp contrib/analysis-extras/lucene-libs/lucene-analyzers-smartcn-7.4.0.jar server/solr-webapp/webapp/WEB-INF/lib

### 、复制新项目的配置文件

cd /root/tar/solr-7.4.0

mkdir server/solr/test

cp -r server/solr/configsets/\_default/conf server/solr/test/

### 、 然后到server/solr/test/conf目录中打开managed-schema文件，增加如下代码

<fieldType name="text\_hmm\_chinese" class="solr.TextField" positionIncrementGap="100">

<analyzer type="index">

<tokenizer class="org.apache.lucene.analysis.cn.smart.HMMChineseTokenizerFactory"/>

</analyzer>

<analyzer type="query">

<tokenizer class="org.apache.lucene.analysis.cn.smart.HMMChineseTokenizerFactory"/>

</analyzer>

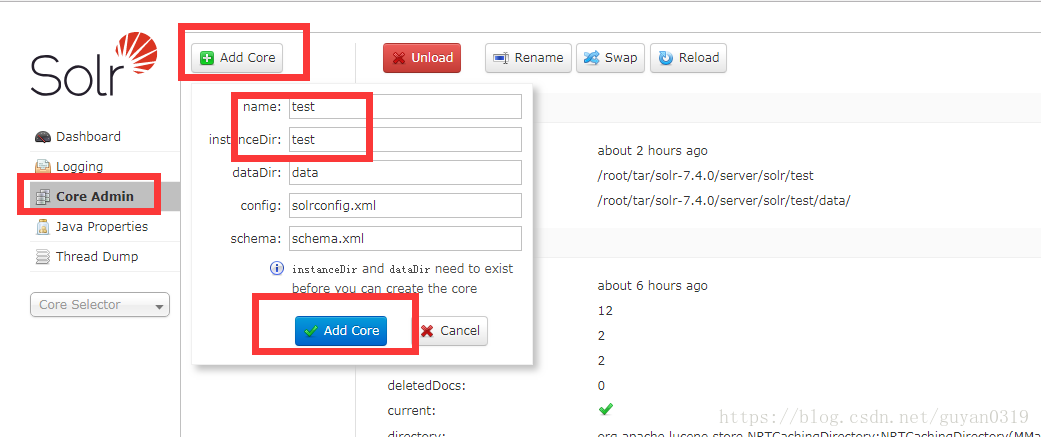
</fieldType>

### 、重启solr

bin/solr restart -force

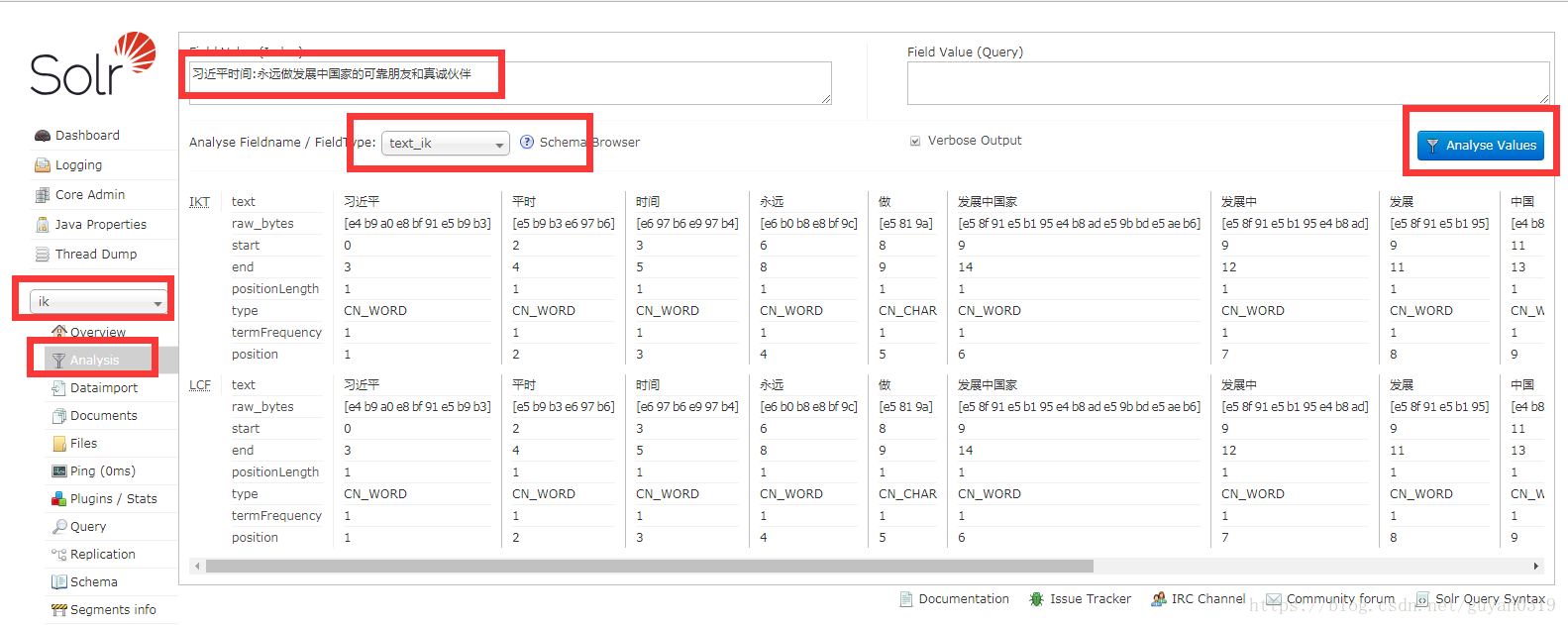
### 、新增项目test

访问项目 http://192.168.3.150:8983

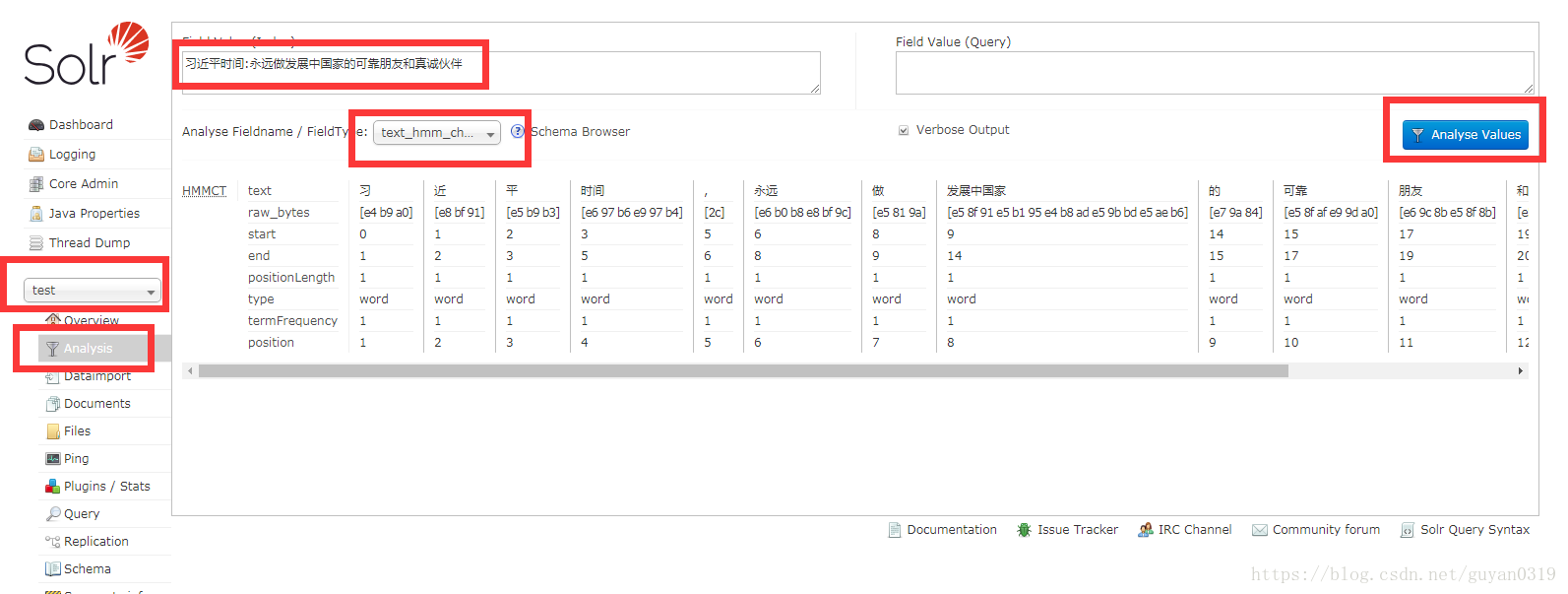


### 对比两个分词的效果

IK分词器



自带分词



个人感觉IK的语义分析更好些.

# 配置业务域

添加文档必须有id域，其他域 必须在solr的schema.xml中定义。

manager-schema.xml中定义

比如：

1. 商品Id 使用schema.xml中的id域
2. 商品标题
3. 商品卖点
4. 商品价格
5. 商品图片
6. 分类名称
7. 商品描述

创建对应的业务域。需要制定中文分析器。

创建步骤：

第一步：指定中文分词器，参考第2 部分。

第二步：配置业务域，type制定使用自定义的FieldType（也就是自己指定的分词器）。

设置业务系统Field

type为string plong的表示不分词。 注意：long类型，在此处为plong类型，不然会报错。

|  |
| --- |
| <field name="item\_title" type="text\_ik" indexed="true" stored="true"/>  <field name="item\_sell\_point" type="text\_ik" indexed="true" stored="true"/>  <field name="item**\_**price" type="plong" indexed="true" stored="true"/>  <field name="item\_image" type="string" indexed="false" stored="true" />  <field name="item\_category\_name" type="string" indexed="true" stored="true" />  <field name="item\_desc" type="text\_ik" indexed="true" stored="false" />  <!—此处使用item\_keywords统括其他四个 -->  <field name="item\_keywords" type="text\_ik" indexed="true" stored="false" multiValued="true"/>  <copyField source="item\_title" dest="item\_keywords"/>  <copyField source="item\_sell\_point" dest="item\_keywords"/>  <copyField source="item\_category\_name" dest="item\_keywords"/>  <copyField source="item\_desc" dest="item\_keywords"/> |

第四步：重启solr

# Solr的Java代码测试（4.x版本）

//solr4创建方式

SolrServer solrServer = new HttpSolrServer(solrUrl);   
//solr5创建方式,在url中指定core名称：core1   
HttpSolrClient solrClient = new HttpSolrClient(solrUrl);   
//solr7创建方式,在url中指定core名称：core1   
HttpSolrClient solrClient = new HttpSolrClient.Builder(solrUrl).build();

## 简单测试- Solr服务连接

**package** cn.test;

**import** java.io.IOException;

**import** java.util.List;

**import** java.util.Map;

**import** org.apache.solr.client.solrj.SolrQuery;

**import** org.apache.solr.client.solrj.SolrServer;

**import** org.apache.solr.client.solrj.SolrServerException;

**import** org.apache.solr.client.solrj.impl.HttpSolrServer;

**import** org.apache.solr.client.solrj.response.QueryResponse;

**import** org.apache.solr.common.SolrDocument;

**import** org.apache.solr.common.SolrDocumentList;

**import** org.apache.solr.common.SolrInputDocument;

**import** org.junit.Test;

**public** **class** SolrTest {

/\*\*

\* 导入索引库

\* **@throws** SolrServerException

\* **@throws** IOException

\*/

@Test

**public** **void** test() **throws** SolrServerException, IOException{

//1 创建SolrServer 建立连接，需要指定url地址

SolrServer solrServer = **new** HttpSolrServer("http://192.168.153.136:8983/solr/ik");

//2 创建SolrInputDocument

SolrInputDocument document = **new** SolrInputDocument();

//3 向文档中添加域

document.addField("id", "test001");

document.addField("item\_title", "设置一个测试数据");

//4 将文档提交到索引库中

solrServer.add(document);

//5提交

solrServer.commit();

}

/\*\*

\* 查询索引

\* **@throws** SolrServerException

\*/

@Test

**public** **void** test2() **throws** SolrServerException{

//1 创建SolrServer 对象

SolrServer solrServer = **new** HttpSolrServer("http://192.168.153.136:8983/solr/ik");

//2 创建SolrQuery对象，设置各种过滤提交 主查询条件 排序

SolrQuery solrQuery = **new** SolrQuery();

//3设置条件

solrQuery.setQuery("阿尔卡特"); //主查询条件

solrQuery.addFilterQuery("item\_price:[0 TO 30000000]"); //过滤条件

solrQuery.set("df", "item\_title"); //默认搜索域

//设置高亮

solrQuery.setHighlight(**true**);//开启高量

solrQuery.setHighlightSimplePre("<em style=\"color:red\">");//高亮前缀

solrQuery.setHighlightSimplePost("</em>");//高亮后缀

solrQuery.addHighlightField("item\_title");//设置高亮显示的域

//4 执行查询

QueryResponse response = solrServer.query(solrQuery);

//5 获取结果集

SolrDocumentList results = response.getResults();

System.***out***.println("查询的总记录数: "+results.getNumFound());

//6遍历结果集

//取高亮

Map<String,Map<String,List<String>>> highLighting = response.getHighlighting();

**for**(SolrDocument temp:results){

//取高亮

List<String> list2 = highLighting.get(temp.get("id")).get("item\_title");

//判断list2 是否为空

String gaoliangstr = "";

**if**(list2 != **null**){

//有高亮

gaoliangstr=list2.get(0);

}**else**{

gaoliangstr = temp.get("item\_title").toString();

}

System.***out***.println(gaoliangstr);

System.***out***.println(temp.getFieldNames());

System.***out***.println(temp.get("id"));

System.***out***.println(temp.get("item\_title"));

}

}

}

## 索引导入：分析setFiled 和 addField 的区别

1 setField 会覆盖其前边的 操作赋值。

document.addField("my\_title;", "mytitle4");

document.addField("my\_title;", "mytitle6");

document.addField("my\_title;", "mytitle5");

document.setField("my\_title;", "mytitle2");

如上操作：结果是 最后的mytitle2 覆盖前边的所有值

"my\_title\_":["mytitle2"],

2 addField 会在前边操作赋值的基础上，追加一个新值。

document.addField("i\_title", "点对点j4");

document.addField("i\_title", "点对点j6");

document.addField("i\_title", "点对点j7");

如上操作：结果是 数据追加在一起

"i\_title":[ "点对点j4",

"点对点j6",

"点对点j7"],

3 此次索引导入的操作，会覆盖以前导入操作的所有值。

导入一次索引，在导入一次。

对相同id的域而言：就像前者的所有值被清空，在写入后者的值。

package cn.test;

import java.io.IOException;

import java.util.List;

import java.util.Map;

import org.apache.solr.client.solrj.SolrQuery;

import org.apache.solr.client.solrj.SolrServer;

import org.apache.solr.client.solrj.SolrServerException;

import org.apache.solr.client.solrj.impl.HttpSolrServer;

import org.apache.solr.client.solrj.response.QueryResponse;

import org.apache.solr.common.SolrDocument;

import org.apache.solr.common.SolrDocumentList;

import org.apache.solr.common.SolrInputDocument;

import org.junit.Test;

public class SolrTest2 {

/\*\*

\* 导入索引练习

\* @throws IOException

\* @throws SolrServerException

\*/

@Test

public void test() throws SolrServerException, IOException{

// 1 创建SolrSever服务连接

SolrServer server = new HttpSolrServer("http://192.168.153.136:8983/solr/ik");

// 2 创建导入文档SolrInputDocument

SolrInputDocument document = new SolrInputDocument();

// 3 设置业务域 在进行索引导入时，必须有id域

document.addField("id", "test");

document.addField("i\_title", "点对点j4");

document.addField("i\_title", "点对点j6");

document.addField("i\_title", "点对点j7");

document.addField("my\_title;", "mytitle4");

document.addField("my\_title;", "mytitle6");

document.addField("my\_title;", "mytitle5");

document.setField("my\_title;", "mytitle2");

/\*\*

\* 结果：

\* response":{"numFound":1,"start":0,"docs":[

{

"id":"test",

"i\_title":[ "点对点j4",

"点对点j6",

"点对点j7"],

"my\_title\_":["mytitle2"],

"\_version\_":1623059029749137408}]

}}

可见addField将在原数据的基础上添加，setField则是覆盖了前边的值，并且其key会多个 ”\_“。

另外，每次导入索引操作，都会覆盖所有以前的值。

\*/

// 4 添加索引文档 服务

server.add(document);

// 5 提交文档

server.commit();

}

@Test

public void test2() throws SolrServerException{

//1 创建连接服务

SolrServer server = new HttpSolrServer("http://192.168.153.136:8983/solr/ik");

//2 创建查询条件

SolrQuery query = new SolrQuery();

//3 设置主查询条件

query.setQuery("\*:\*"); //设置主查询条件

query.addFilterQuery("id:test"); //过滤条件

query.set("df", "item\_title"); //默认搜索域

//4 执行查询

QueryResponse response = server.query(query);

//5 获取结果集

SolrDocumentList results = response.getResults();

//6 遍历结果集

for(SolrDocument temp : results){

System.out.println(temp.get("id"));

System.out.println(temp.get("item\_title\_"));

}

}

}

## 索引查询：SolrQuery SolrDocument

设置主查询条件：query.setQuery(“”)

设置过滤条件：query.addFilterQuery();

设置默认搜索域：query.set(“df”，““);

设置分页：query.setStart()

query.setRows()

/\*\*

\* 查询索引

\* **@throws** SolrServerException

\*/

**public** **void** test2(Integer page ,Integer rows) **throws** SolrServerException{

//1 创建SolrServer 对象

SolrServer solrServer = **new** HttpSolrServer("http://192.168.153.136:8983/solr/ik");

//2 创建SolrQuery对象，设置各种过滤提交 主查询条件 排序

SolrQuery solrQuery = **new** SolrQuery();

//3 设置查询条件

//3.1 设置主查询条件

solrQuery.setQuery("阿尔卡特");

//3.2 设置过滤条件

solrQuery.addFilterQuery("item\_price:[0 TO 30000000]");

//3.3 设置过滤条件 设置分页

**if**(page == **null**)page=1;

**if**(rows == **null**)rows=60;

solrQuery.setStart((page-1)\*rows);

solrQuery.setRows(rows);

//3.4 设置默认搜索域

solrQuery.set("df", "item\_title");

//3.5 设置高亮

solrQuery.setHighlight(**true**);//开启高量

solrQuery.setHighlightSimplePre("<em style=\"color:red\">");//高亮前缀

solrQuery.setHighlightSimplePost("</em>");//高亮后缀

solrQuery.addHighlightField("item\_title");//设置高亮显示的域

//4 执行查询

QueryResponse response = solrServer.query(solrQuery);

//5 获取结果集

SolrDocumentList results = response.getResults();

System.***out***.println("查询的总记录数: "+results.getNumFound());

//6遍历结果集

-默认搜索域//取高亮

Map<String,Map<String,List<String>>> highLighting = response.getHighlighting();

**for**(SolrDocument temp:results){

//取高亮

List<String> list2 = highLighting.get(temp.get("id")).get("item\_title");

//判断list2 是否为空

String gaoliangstr = "";

**if**(list2 != **null**){

//有高亮

gaoliangstr=list2.get(0);

}**else**{

gaoliangstr = temp.get("item\_title").toString();

}

System.***out***.println(gaoliangstr);

System.***out***.println(temp.getFieldNames());

System.***out***.println(temp.get("id"));

System.***out***.println(temp.get("item\_title"));

}

}

# SolrCloud （集群）安装

## solrcloud简介

### 什么是solrcloud？

SolrCloud(solr 云)是Solr提供的分布式搜索方案，当你需要大规模，容错，分布式索引和检索能力时使用 SolrCloud当一个系统的索引数据量少的时候是不需要使用SolrCloud的，当索引量很大，搜索请求并发很高，这时需要使用SolrCloud来满足这些需求

### solrcloud的结构



## **搭建准备工作**

### 准备java环境

zookeeper是用Java编写的，运行在Java环境上，因此，在部署zk的机器上需要安装Java运行环境。安装JDK

### solr7.6.0下载

http://www.apache.org/dyn/closer.lua/lucene/solr/7.6.0

### 安装并配置好一个solr单机版。

见前边的内容。此时安装好了一个单机版solr在/usr/local/solr文件夹下。

## 伪solr集群搭建（一台服务器）

### 先进行zookeeper集群搭建

详情见zookeeper集群搭建手册

### SolrCloud搭建

1 在/usr/local/下创建文件夹solrCloud

[root@localhost local]# mkdir solrCloud

2 在solrCloud下创建四个文件夹

[root@localhost solrCloud]# mkdir solr01

[root@localhost solrCloud]# mkdir solr02

[root@localhost solrCloud]# mkdir solr03

[root@localhost solrCloud]# mkdir solr04

[root@localhost solrCloud]# ll

total 0

drwxr-xr-x 2 root root 6 Jan 19 19:54 solr01

drwxr-xr-x 2 root root 6 Jan 19 19:55 solr02

drwxr-xr-x 2 root root 6 Jan 19 19:55 solr03

drwxr-xr-x 2 root root 6 Jan 19 19:55 solr04

3 复制安装完成的单机版Solr到这四个文件夹下

[root@localhost solr]# ll

total 0

drwxr-xr-x 9 root root 201 Jan 17 12:22 solr-7.6.0

[root@localhost solr]# cp -r solr-7.6.0/ /usr/local/solrCloud/solr01

[root@localhost solr]# cp -r solr-7.6.0/ /usr/local/solrCloud/solr02

[root@localhost solr]# cp -r solr-7.6.0/ /usr/local/solrCloud/solr03

[root@localhost solr]# cp -r solr-7.6.0/ /usr/local/solrCloud/solr04

4 ：solr01、solr02、solr03、solr04 修改solr.in.sh（solr/bin/solr.in.sh）配置

[root@localhost solrCloud]# vim solr01/solr-7.6.0/bin/solr.in.sh

[root@localhost solrCloud]# vim solr02/solr-7.6.0/bin/solr.in.sh

[root@localhost solrCloud]# vim solr03/solr-7.6.0/bin/solr.in.sh

[root@localhost solrCloud]# vim solr04/solr-7.6.0/bin/solr.in.sh

#让solr关联zookeeper

ZK\_HOST="192.168.153.136:2182,192.168.153.136:2183,192.168.153.136:2184"

#solr主机ip地址

SOLR\_HOST="192.168.153.136"

#更改端口分别为8984，solr2端口为8985，solr6端口为8986，solr4端口为8987

SOLR\_PORT=8984

3：solr1、solr2、solr3、solr4启动，停止脚本

▶▶启动    solr\_all\_start.sh 别忘了添加执行权限

[root@localhost solrCloud]# vim solr\_all\_start.sh

/usr/local/solrCloud/solr01/solr-7.6.0/bin/solr start -force

/usr/local/solrCloud/solr02/solr-7.6.0/bin/solr start -force

/usr/local/solrCloud/solr03/solr-7.6.0/bin/solr start -force

/usr/local/solrCloud/solr04/solr-7.6.0/bin/solr start -force

▶▶ 停止   solr\_all\_stop.sh

/usr/local/solrCloud/solr01/solr-7.6.0/bin/solr stop

/usr/local/solrCloud/solr02/solr-7.6.0/bin/solr stop

/usr/local/solrCloud/solr03/solr-7.6.0/bin/solr stop

/usr/local/solrCloud/solr04/solr-7.6.0/bin/solr stop

### **启动zk集群、solrcloud，创建collection**

./zk\_start\_all.sh #启动zookeeper集群

./solr\_all\_start.sh #启动solrcloud

cd /usr/local/solrCloud/solr01/solr-7.6.0/bin #进入solrcloud其中任意一个节点solr的bin目录

#-c 实例名称 -s 分片个数、-rf副本个数 (创建2个分片，2个副本，-c MySolrCloud 实例名称 ，-s，-rf默认都是1，即不分片，无副本)

./solr create -c MySolrCloud -s 2 -rf 2 -force #创建集合，指定2个分片、2个副本

#### 启动zk集群

[root@localhost zookeepercluster]# ./zookeeper\_start\_all.sh

ZooKeeper JMX enabled by default

Using config: /usr/local/zookeepercluster/zookeeper01/zookeeper-3.4.13/bin/../conf/zoo.cfg

Starting zookeeper ... STARTED

ZooKeeper JMX enabled by default

Using config: /usr/local/zookeepercluster/zookeeper02/zookeeper-3.4.13/bin/../conf/zoo.cfg

Starting zookeeper ... STARTED

ZooKeeper JMX enabled by default

Using config: /usr/local/zookeepercluster/zookeeper03/zookeeper-3.4.13/bin/../conf/zoo.cfg

Starting zookeeper ... STARTED

#### 启动SolrCloud

[root@localhost solrCloud]# ./solr\_all\_start.sh

\*\*\* [WARN] \*\*\* Your open file limit is currently 1024.

It should be set to 65000 to avoid operational disruption.

If you no longer wish to see this warning, set SOLR\_ULIMIT\_CHECKS to false in your profile or solr.in.sh

\*\*\* [WARN] \*\*\* Your Max Processes Limit is currently 3764.

It should be set to 65000 to avoid operational disruption.

If you no longer wish to see this warning, set SOLR\_ULIMIT\_CHECKS to false in your profile or solr.in.sh

Waiting up to 180 seconds to see Solr running on port 8984 [-]

Started Solr server on port 8984 (pid=47670). Happy searching!

\*\*\* [WARN] \*\*\* Your open file limit is currently 1024.

It should be set to 65000 to avoid operational disruption.

If you no longer wish to see this warning, set SOLR\_ULIMIT\_CHECKS to false in your profile or solr.in.sh

\*\*\* [WARN] \*\*\* Your Max Processes Limit is currently 3764.

It should be set to 65000 to avoid operational disruption.

If you no longer wish to see this warning, set SOLR\_ULIMIT\_CHECKS to false in your profile or solr.in.sh

Waiting up to 180 seconds to see Solr running on port 8985 [-]

Started Solr server on port 8985 (pid=47840). Happy searching!

\*\*\* [WARN] \*\*\* Your open file limit is currently 1024.

It should be set to 65000 to avoid operational disruption.

If you no longer wish to see this warning, set SOLR\_ULIMIT\_CHECKS to false in your profile or solr.in.sh

\*\*\* [WARN] \*\*\* Your Max Processes Limit is currently 3764.

It should be set to 65000 to avoid operational disruption.

If you no longer wish to see this warning, set SOLR\_ULIMIT\_CHECKS to false in your profile or solr.in.sh

Waiting up to 180 seconds to see Solr running on port 8986 [/]

Started Solr server on port 8986 (pid=48003). Happy searching!

\*\*\* [WARN] \*\*\* Your open file limit is currently 1024.

It should be set to 65000 to avoid operational disruption.

If you no longer wish to see this warning, set SOLR\_ULIMIT\_CHECKS to false in your profile or solr.in.sh

\*\*\* [WARN] \*\*\* Your Max Processes Limit is currently 3764.

It should be set to 65000 to avoid operational disruption.

If you no longer wish to see this warning, set SOLR\_ULIMIT\_CHECKS to false in your profile or solr.in.sh

Waiting up to 180 seconds to see Solr running on port 8987 [|]

Started Solr server on port 8987 (pid=48183). Happy searching!

#### 创建Collection

[root@localhost bin]# ./solr create -c MySolrCloud -s 2 -rf 2 -force

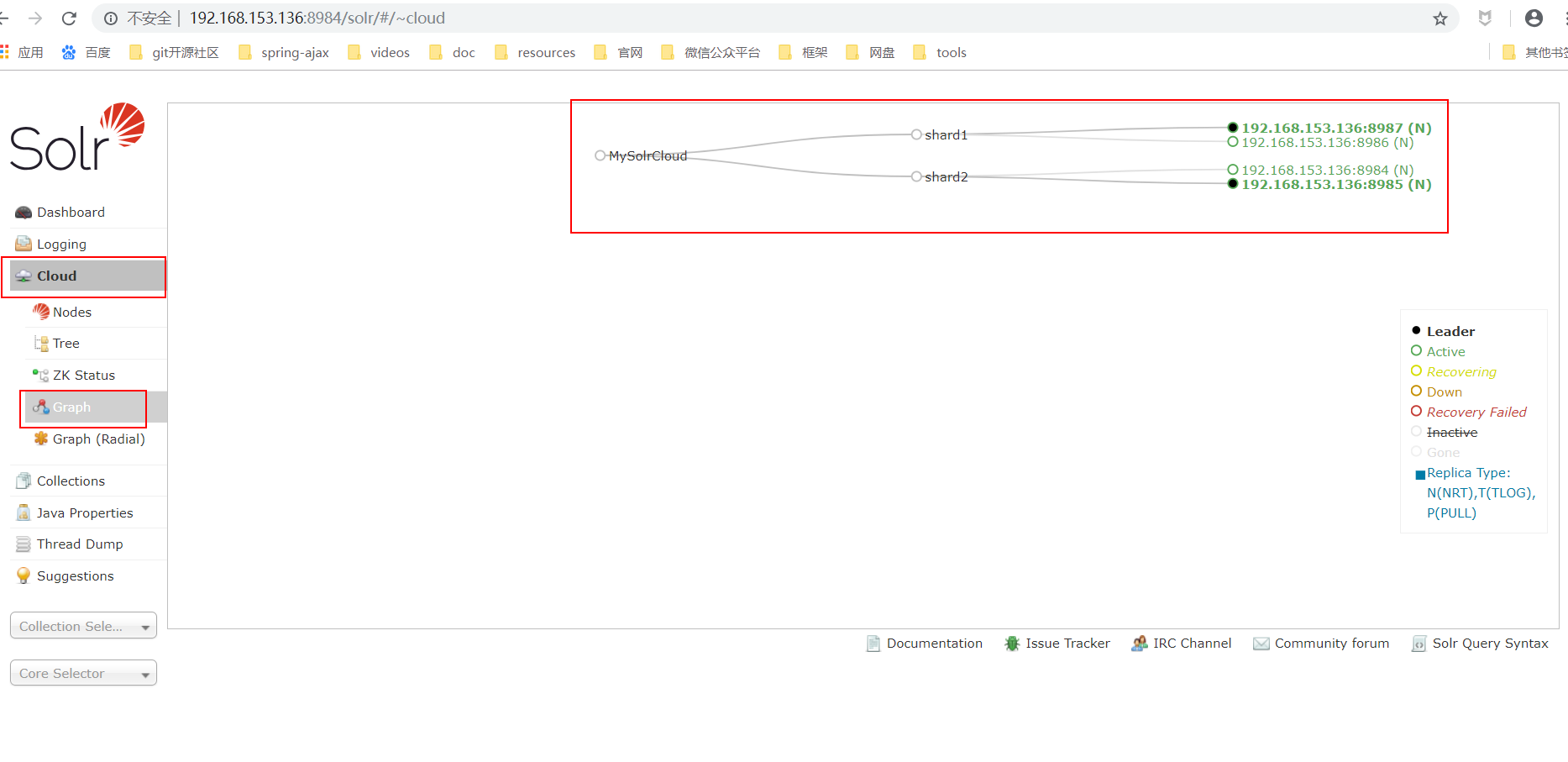
WARNING: Using \_default configset with data driven schema functionality. NOT RECOMMENDED for production use.

To turn off: bin/solr config -c MySolrCloud -p 8987 -action set-user-property -property update.autoCreateFields -value false

INFO - 2019-01-19 20:30:53.335; org.apache.solr.util.configuration.SSLCredentialProviderFactory; Processing SSL Credential Provider chain: env;sysprop

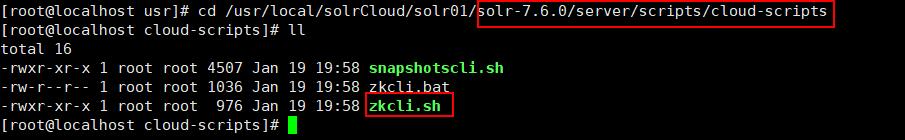
Created collection 'MySolrCloud' with 2 shard(s), 2 replica(s) with config-set 'MySolrCloud'

访问solrcloud任意一个节点服务器，查看collection集群



### **上传solr配置文件到zookeeper，统一管理**

使用 solr下的zkcli.sh 命令上传 ， solr/server/scripts/cloud-scripts目录下



#zkhost：指定zookeeper集群

#-cmd upconfig 上传配置

#condir: 指定要上传的文件

#confname: 命名上传zk之后的文件

./zkcli.sh -zkhost 192.168.153.136:2182,192.168.153.136:2183,192.168.153.136:2184 -cmd upconfig -confdir /usr/local/solrCloud/solr01/solr-7.6.0/server/solr/configsets/\_default/conf/ -confname MySolrConfig

### 查看上传zk的MySolrConfig,进入任意zookeeper

查看zookeeper上的配置文件：

使用zookeeper目录下的bin/zkCli.sh命令查看zookeeper上的配置文件：

注意点(如果不是默认的端口2181，要连接使用命令：

./zkCli.sh -server 192.168.153.136:2182

解释：

-server :指定ip地址

：2181 ：指定连接的端口。

[root@localhost bin]# ./zkCli.sh -server 192.168.153.136:2182

Connecting to 192.168.153.136:2182

2019-01-19 21:40:29,786 [myid:] - INFO [main:Environment@100] - Client environment:zookeeper.version=3.4.13-2d71af4dbe22557fda74f9a9b4309b15a7487f03, built on 06/29/2018 04:05 GMT

2019-01-19 21:40:29,790 [myid:] - INFO [main:Environment@100] - Client environment:host.name=localhost

2019-01-19 21:40:29,790 [myid:] - INFO [main:Environment@100] - Client environment:java.version=1.8.0\_181

2019-01-19 21:40:29,795 [myid:] - INFO [main:Environment@100] - Client environment:java.vendor=Oracle Corporation

2019-01-19 21:40:29,795 [myid:] - INFO [main:Environment@100] - Client environment:java.home=/usr/java/jdk1.8.0\_181/jre

2019-01-19 21:40:29,795 [myid:] - INFO [main:Environment@100] - Client environment:java.class.path=/usr/local/zookeepercluster/zookeeper01/zookeeper

……

Welcome to ZooKeeper!

JLine support is enabled

2019-01-19 21:40:29,982 [myid:] - INFO [main-SendThread(192.168.153.136:2182):ClientCnxn$SendThread@1029] - Opening socket connection to server 192.168.153.136/192.168.153.136:2182. Will not attempt to authenticate using SASL (unknown error)

2019-01-19 21:40:29,992 [myid:] - INFO [main-SendThread(192.168.153.136:2182):ClientCnxn$SendThread@879] - Socket connection established to 192.168.153.136/192.168.153.136:2182, initiating session

2019-01-19 21:40:30,020 [myid:] - INFO [main-SendThread(192.168.153.136:2182):ClientCnxn$SendThread@1303] - Session establishment complete on server 192.168.153.136/192.168.153.136:2182, sessionid = 0x1000a68d9c40008, negotiated timeout = 30000

WATCHER::

WatchedEvent state:SyncConnected type:None path:null

[zk: 192.168.153.136:2182(CONNECTED) 0] ls /

[configs, zookeeper, overseer, aliases.json, live\_nodes, collections, overseer\_elect, security.json, clusterstate.json, autoscaling, autoscaling.json]

[zk: 192.168.153.136:2182(CONNECTED) 1] ls /configs/MySolrConfig

[managed-schema, protwords.txt, solrconfig.xml, synonyms.txt, stopwords.txt, lang, params.json]

## solr集群搭建(未测试)

思路：先在一台服务器上搭建好zookeeper、solr，然后使用虚拟机克隆复制两份，搭建成集群。

### 安装zookeeper

1：解压、重命名

tar -zxvf zookeeper-3.4.12.tar.gz

mv zookeeper-3.4.12 zookeeper

2：zookeeper目录下新建数据目录data、日志目录log

cd zookeeper

mkdir data

mkdir log

3：zookeeper/data目录下创建zookeeper标识文件 myid，并设置标识id

cd zookeeper/data/

touch myid

echo 1 > myid    #设置zookeeper的标识id为1

4：创建zookeeper的配置文件

cd zookeeper/conf

mv zoo\_sample.cfg zoo.cfg

5：修改zookeeper配置文件zoo.cfg

▶▶ 修改dataDir和clientPort参数

# 配置zookeeper数据目录

dataDir=/usr/local/soft/zookeeper/data

# 配置zookeeper端口

clientPort=2181

▶▶ 添加dataLogDir和server.id参数

# 配置zookeeper的日志目录

dataLogDir=/usr/local/soft/zookeeper/log

# 配置zookeeper集群   server.标识id = ip：通信端口：投票选举端口

server.1=192.168.1.211:2881:3881

server.2=192.168.1.212:2882:3882

server.3=192.168.1.213:2883:3883

### solr安装

1：解压、重命名

tar -zxvf solr-7.5.0.tgz

mv solr-7.5.0 solr

2：修改solr.in.sh（solr/bin/solr.in.sh）配置

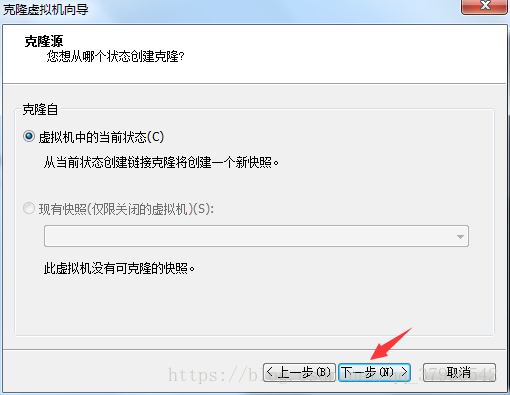
#让solr关联zookeeper

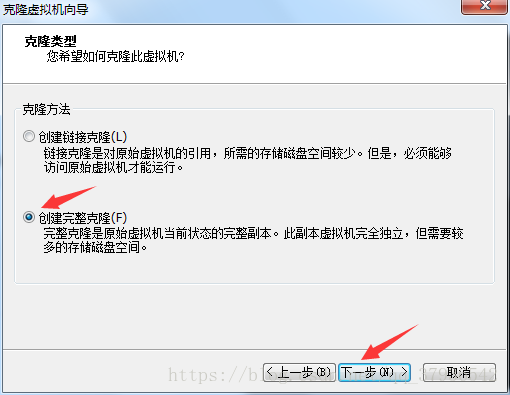
ZK\_HOST="192.168.1.211:2181,192.168.1.212:2181,192.168.1.213:2181"

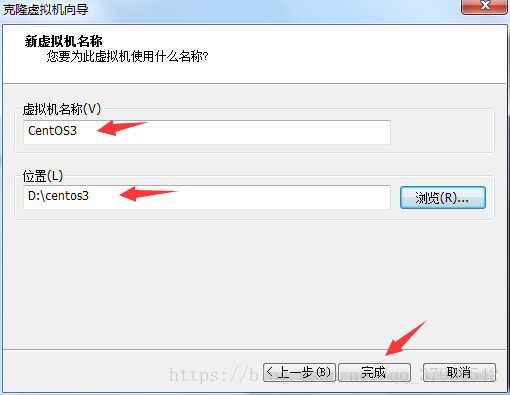
### 复制虚拟机

1：先关闭虚拟机，选中要克隆的虚拟机，右击->管理->克隆，按图操作









2：修改两台服务器的IP  和 zookeeper的标识ID

▶▶ 修改服务器的IP分别为

192.168.1.212

192.168.1.213

▶▶ 修改zookeeper的标识ID

修改192.168.1.212的myid文件内容为2

修改192.168.1.213的myid文件内容为3

☛ 创建collection ☚

1：启动三台服务器的zookeeper

2：启动三台服务器的solr

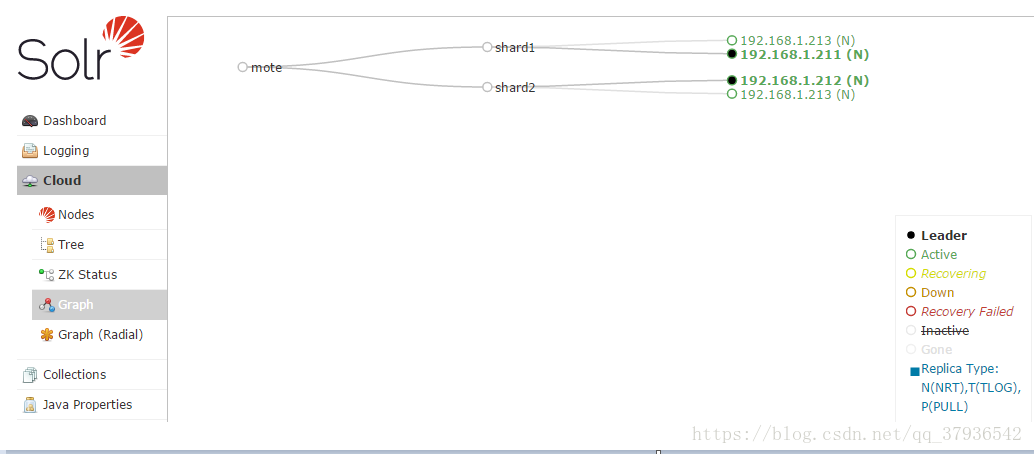
3：选择任意一个服务器的solr

cd solr/bin/        #进入solrcloud其中任意一个节点solr的bin目录

#-s 分片个数、-rf副本个数

./solr create -c mote -s 2 -rf 2 -force   #创建集合，指定2个分片、2个副本

4：访问solrcloud任意一个节点服务器，查看collection集群



5：上传solr配置文件到zookeeper

使用 zkcli.sh（solr/server/scripts/cloud-scripts/） 命令上传

#zkhost：指定zookeeper集群

#condir: 指定要上传的文件

#confname: 命名上传zk之后的文件

./zkcli.sh -zkhost 192.168.1.211:2181,192.168.1.212:2181,192.168.1.213:2181 -cmd upconfig -confdir /usr/local/soft/solr/server/solr/configsets/\_default/conf/ -confname solrconfig

## URL创建和删除Collection

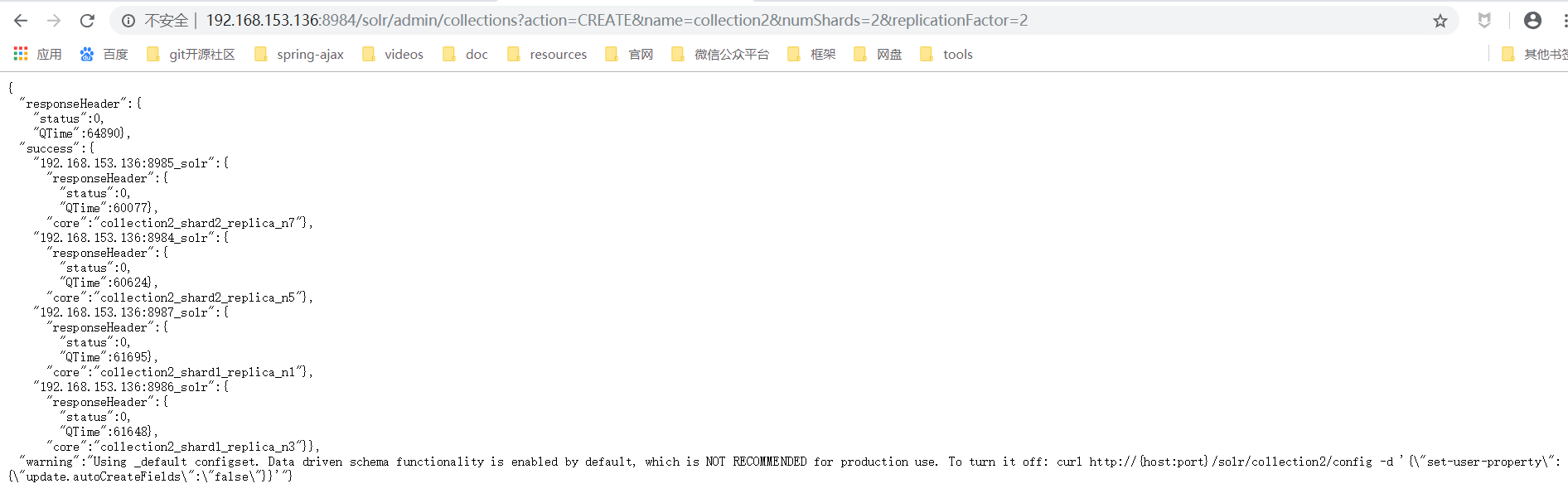
### 创建新的Collection进行分片处理。

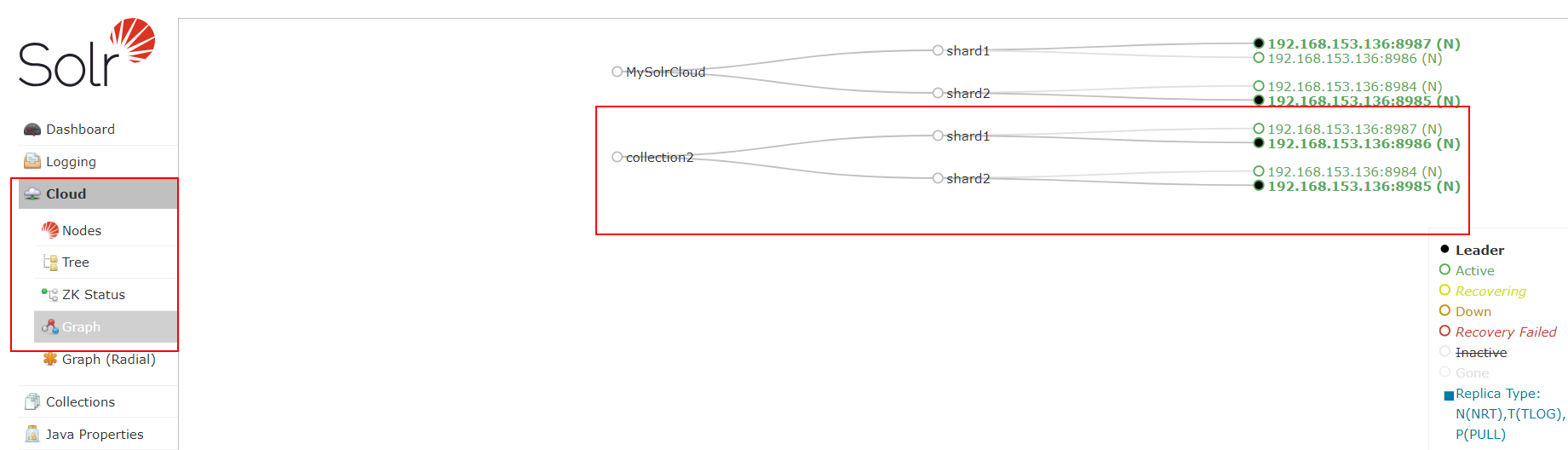
<http://192.168.153.136:8984/solr/admin/collections?action=CREATE&name=collection2&numShards=2&replicationFactor=2>

name=collection2 是collection名称

numShards=2 是两片

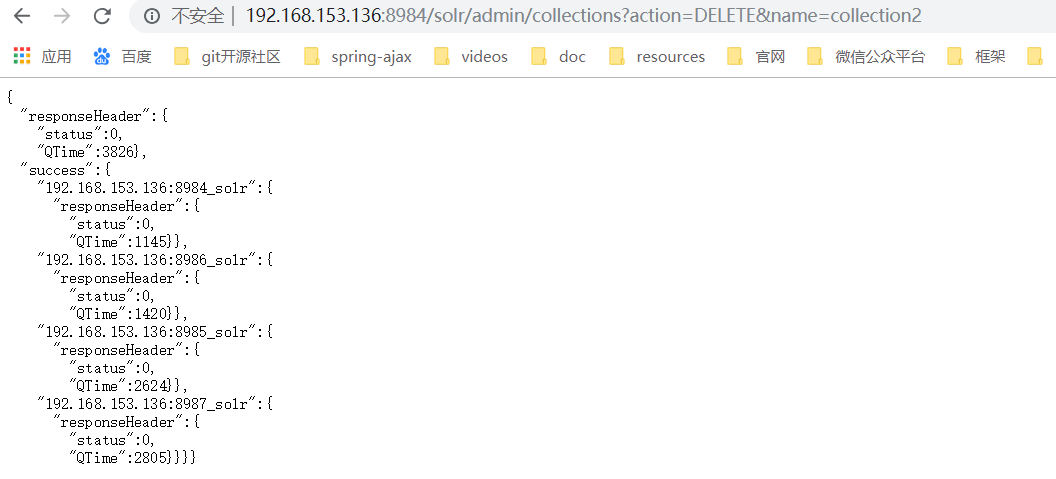
replicationFactory=2 是两个备份

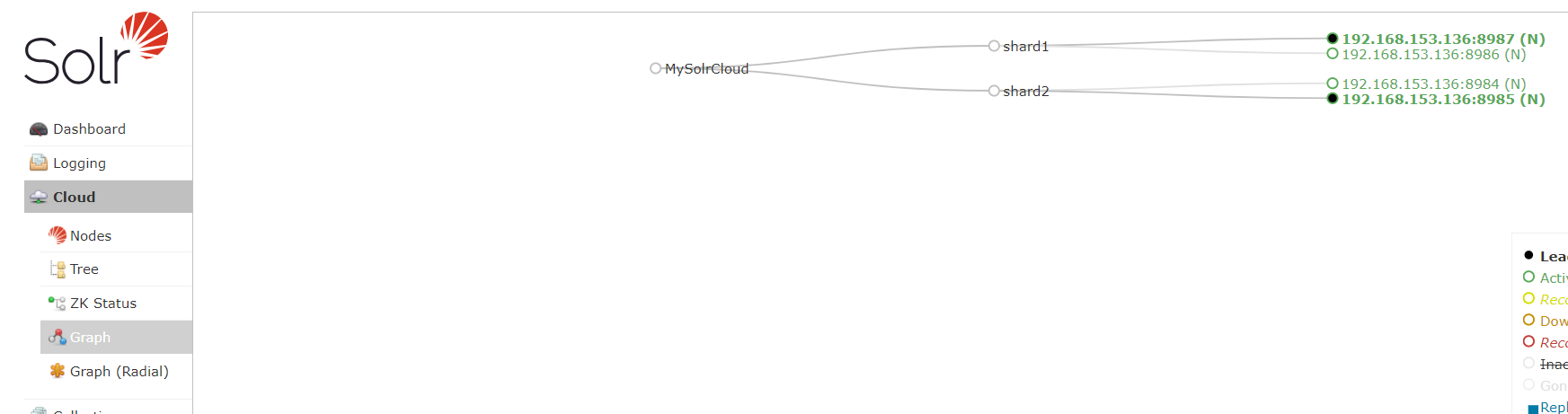




### 删除不用的Collection

<http://192.168.153.136:8984/solr/admin/collections?action=DELETE&name=collection2>





# solr7.x 版本代码测试

版本不同，获取客户端的方式也不相同：

//solr4创建方式

SolrServer solrServer = new HttpSolrServer(solrUrl);   
//solr5创建方式,在url中指定core名称：core1   
HttpSolrClient solrClient = new HttpSolrClient(solrUrl);   
//solr7创建方式,在url中指定core名称：core1   
HttpSolrClient solrClient = new HttpSolrClient.Builder(solrUrl).build();

## 单机版测试

**package** cn.test;

**import** java.io.IOException;

**import** java.util.List;

**import** java.util.Map;

**import** org.apache.solr.client.solrj.SolrClient;

**import** org.apache.solr.client.solrj.SolrQuery;

**import** org.apache.solr.client.solrj.SolrServerException;

**import** org.apache.solr.client.solrj.impl.HttpSolrClient;

**import** org.apache.solr.client.solrj.response.QueryResponse;

**import** org.apache.solr.common.SolrDocument;

**import** org.apache.solr.common.SolrDocumentList;

**import** org.apache.solr.common.SolrInputDocument;

**import** org.junit.Test;

**public** **class** SolrTest {

/\*\*

\* 导入索引库

\* **@throws** SolrServerException

\* **@throws** IOException

\*/

@Test

**public** **void** test() **throws** SolrServerException, IOException{

//1 创建SolrServer 建立连接，需要指定url地址

SolrClient client = **new** HttpSolrClient.Builder("http://192.168.153.136:8983/solr/ik").build();

//2 创建SolrInputDocument

SolrInputDocument document = **new** SolrInputDocument();

//3 向文档中添加域 添加时必须有id域

document.addField("id", "test003");

document.addField("item\_title", "设置一个测试数据");

//4 将文档提交到索引库中

client.add(document);

//5提交

client.commit();

}

/\*\*

\* 查询索引

\* **@throws** SolrServerException

\* **@throws** IOException

\*/

@Test

**public** **void** test2() **throws** SolrServerException, IOException{

//1 创建SolrServer 对象

SolrClient solrServer = **new** HttpSolrClient.Builder("http://192.168.153.136:8983/solr/ik").build();

//2 创建SolrQuery对象，设置各种过滤提交 主查询条件 排序

SolrQuery solrQuery = **new** SolrQuery();

//3设置条件

solrQuery.setQuery("阿尔卡特"); //主查询条件

solrQuery.addFilterQuery("item\_price:[0 TO 30000000]"); //过滤条件

solrQuery.set("df", "item\_title"); //默认搜索域

//设置高亮

solrQuery.setHighlight(**true**);//开启高量

solrQuery.setHighlightSimplePre("<em style=\"color:red\">");//高亮前缀

solrQuery.setHighlightSimplePost("</em>");//高亮后缀

solrQuery.addHighlightField("item\_title");//设置高亮显示的域

//4 执行查询

QueryResponse response = solrServer.query(solrQuery);

//5 获取结果集

SolrDocumentList results = response.getResults();

System.***out***.println("查询的总记录数: "+results.getNumFound());

//6遍历结果集

//取高亮

Map<String,Map<String,List<String>>> highLighting = response.getHighlighting();

**for**(SolrDocument temp:results){

//取高亮

List<String> list2 = highLighting.get(temp.get("id")).get("item\_title");

//判断list2 是否为空

String gaoliangstr = "";

**if**(list2 != **null**){

//有高亮

gaoliangstr=list2.get(0);

}**else**{

gaoliangstr = temp.get("item\_title").toString();

}

System.***out***.println(gaoliangstr);

System.***out***.println(temp.getFieldNames());

System.***out***.println(temp.get("id"));

System.***out***.println(temp.get("item\_title"));

}

}

}

**package** cn.test;

**import** java.io.IOException;

**import** java.util.List;

**import** java.util.Map;

**import** org.apache.solr.client.solrj.SolrClient;

**import** org.apache.solr.client.solrj.SolrQuery;

**import** org.apache.solr.client.solrj.SolrServerException;

**import** org.apache.solr.client.solrj.impl.HttpSolrClient;

**import** org.apache.solr.client.solrj.response.QueryResponse;

**import** org.apache.solr.common.SolrDocument;

**import** org.apache.solr.common.SolrDocumentList;

**import** org.apache.solr.common.SolrInputDocument;

**import** org.junit.Test;

**public** **class** SolrTest2 {

/\*\*

\* 导入索引练习

\* **@throws** IOException

\* **@throws** SolrServerException

\*/

@Test

**public** **void** test() **throws** SolrServerException, IOException{

// 1 创建SolrSever服务连接

SolrClient server = **new** HttpSolrClient.Builder("http://192.168.153.136:8983/solr/ik").build();

// 2 创建导入文档SolrInputDocument

SolrInputDocument document = **new** SolrInputDocument();

// 3 设置业务域 在进行索引导入时，必须有id域

document.addField("id", "test");

document.addField("i\_title", "点对点j4");

document.addField("i\_title", "点对点j6");

document.addField("i\_title", "点对点j7");

document.addField("my\_title;", "mytitle4");

document.addField("my\_title;", "mytitle6");

document.addField("my\_title;", "mytitle5");

document.setField("my\_title;", "mytitle2");

/\*\*

\* 结果：

\* response":{"numFound":1,"start":0,"docs":[

{

"id":"test",

"i\_title":[ "点对点j4",

"点对点j6",

"点对点j7"],

"my\_title\_":["mytitle2"],

"\_version\_":1623059029749137408}]

}}

可见addField将在原数据的基础上添加，setField则是覆盖了前边的值，并且其key会多个 ”\_“。

另外，每次导入索引操作，都会覆盖所有以前的值。

\*/

// 4 添加索引文档 服务

server.add(document);

// 5 提交文档

server.commit();

}

@Test

**public** **void** test2() **throws** SolrServerException, IOException{

//1 创建连接服务

SolrClient server = **new** HttpSolrClient.Builder("http://192.168.153.136:8983/solr/ik").build();

//2 创建查询条件

SolrQuery query = **new** SolrQuery();

//3 设置主查询条件

query.setQuery("\*:\*"); //设置主查询条件

query.addFilterQuery("id:test"); //过滤条件

query.set("df", "item\_title"); //默认搜索域

//4 执行查询

QueryResponse response = server.query(query);

//5 获取结果集

SolrDocumentList results = response.getResults();

//6 遍历结果集

**for**(SolrDocument temp : results){

System.***out***.println(temp.get("id"));

System.***out***.println(temp.get("item\_title\_"));

}

}

}

## SolrCloud测试

package cn.test;

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;

import java.util.Optional;

import org.apache.solr.client.solrj.SolrQuery;

import org.apache.solr.client.solrj.SolrServerException;

import org.apache.solr.client.solrj.impl.CloudSolrClient;

import org.apache.solr.client.solrj.response.QueryResponse;

import org.apache.solr.common.SolrDocument;

import org.apache.solr.common.SolrDocumentList;

import org.apache.solr.common.SolrInputDocument;

import org.junit.Test;

public class SolrCluster {

/\*\*

\* 导入索引

\* @throws SolrServerException

\* @throws IOException

\*/

@Test

public void test() throws SolrServerException, IOException {

//1 zookeeper地址集合

List<String> zkServers = new ArrayList<String>();

zkServers.add("192.168.153.136:2182");

zkServers.add("192.168.153.136:2183");

zkServers.add("192.168.153.136:2184");

//2 solr集群客户端 注意：第二个参数：Optional.of("/") 可选择有/solr /infra-solr 和 / 如果zkHost设置时，后边没有跟/solr等，就使用/

//不然会报错：Cannot connect to cluster at 192.168.153.136:2182,192.168.153.136:2183,192.168.153.136:2183: cluster not found/not ready

CloudSolrClient client = new CloudSolrClient.Builder(zkServers, Optional.of("/")).build();

//3 设置默认的集群

client.setDefaultCollection("MySolrCloud");

//4 输入文档

SolrInputDocument doc = new SolrInputDocument();

//6 业务域设置

doc.addField("id", "SolrCloudTest");

doc.addField("item\_title", "测试数据");

//7 添加文档到服务区

client.add(doc);

//8 提交服务

client.commit();

//9 关闭客户端

client.close();

System.out.println("导入完毕");

}

/\*\*

\* 搜索索引

\* @throws SolrServerException

\* @throws IOException

\*/

@Test

public void test2() throws SolrServerException, IOException {

//1 zookeeper地址集合

List<String> zkServers = new ArrayList<String>();

zkServers.add("192.168.153.136:2182");

zkServers.add("192.168.153.136:2183");

zkServers.add("192.168.153.136:2184");

//2 solr集群客户端 注意：第二个参数：Optional.of("/") 可选择有/solr /infra-solr 和 / 如果zkHost设置时，后边没有跟/solr等，就使用/

//不然会报错：Cannot connect to cluster at 192.168.153.136:2182,192.168.153.136:2183,192.168.153.136:2183: cluster not found/not ready

CloudSolrClient client = new CloudSolrClient.Builder(zkServers, Optional.of("/")).build();

//3 设置默认的集群

client.setDefaultCollection("MySolrCloud");

SolrQuery query = new SolrQuery();

query.setQuery("测试数据"); //主查询条件

query.set("df", "item\_title"); //默认搜索域

QueryResponse result = client.query(query);

SolrDocumentList results = result.getResults();

for(SolrDocument temp :results){

System.out.println(temp.getFieldNames());

System.out.println(temp.getFieldValue("item\_title"));

System.out.println(temp.getFieldValue("id"));

}

//9 关闭客户端

client.close();

System.out.println("查询完毕");

}

}

# solr节点down掉恢复

1 检查down掉机器的分片文件夹（usr/local/solr-cores）中数据是否还在，如果还在，把每个分片下data/index下的write.lock文件删除，如果不在就可以从其他数据库移动分片数据过来，删除write.lock文件以及data/tlog下的tlog文件。或者也可以重新建新的分片。（本文以分片文件还在为例）

2 检查修改solr.xml

检查usr/local/solr-cores下solr.xml文件是否正确，可以对比其他正常分片的solr.xml文件。注意分片、集群的配置数据。

3 检查数据所属组

完成之后检查分片数据文件夹及文件所属用户，如果不是solr，修改为solr。

4 重新启动solr节点

# solr更新schema.xml后，重新把配置文件注入到zookeeper：

./zkcli.sh -z Master.Hadoop:2181 -cmd putfile /solr/configs/poi\_index\_update/schema.xml

/data/solr-5.3.1/server/solr/configsets/poi\_index\_update\_configs/conf/schema.xml

-- 前一个文件为zookeeper文件路径，后面为本地文件路径

http://localhost:8983/solr/admin/cores?action=RELOAD&core=core0 重新加载这个core，一般通过浏览器界面操作。