# solr单机版



把solr压缩包上传到服务器并解压

## 创建solrhome

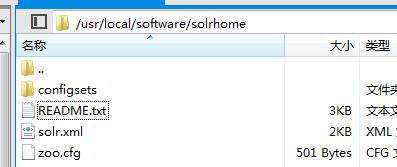
在/usr/local/software创建solrhome文件夹，每个solr服务都有一个自己的solrhome,里面存放了solr节点core的相关信息。

将solr-5.5.5/server/solr/下所有文件复制到solrhome

复制的命令是：

cp -rf /usr/local/software/solr-5.5.5/server/solr/\* /usr/local/software/solrhome





## 搭建solr服务

说明：solr的工程需要部署到tomcat下，但是5.5.5的版本中没有可以直接使用的war包，所以需要自己配置资源。参考资料中提供了可以直接使用的solr.war。将solr.war上传到tomcat的webapp目录，启动tomcat就可以运行。不同的虚拟机上可能要修改solrhome的配置。启动tomcat后会解压solr.war。如果你要修改配置，需要停止tomcat，删除solr.war，再解压后的solr文件夹中修改。否则重启tomcat后solr.war会重新解压，覆盖掉你修改的配置。如果你使用了solr.war，可以跳过本节的内容。

在解压后的solr-5.5.5中：

solr-5.5.5/server/solr-webapp/webapp这个是solr服务工程的模板，只是其中还缺少一些资源，参考资料中的solr.war就是使用这个模板配置好的工程。

进入解压后的目录。把solr-5.5.5/server/lib/ext下所有的jar包复制到solr-5.5.5/server/solr-webapp/webapp/WEB-INF/lib



将solr-5.5.5/dist下的solr-dataimporthandler-5.5.5.jar和solr-dataimporthandler-extras-5.5.5.jar放到solr-5.5.5/server/solr-webapp/webapp/WEB-INF/lib

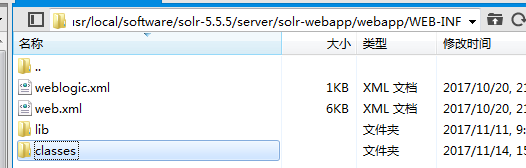


复制的命令是：

cp /usr/local/software/solr-5.5.5/dist/solr-dataimporthandler-5.5.5.jar /usr/local/software/solr-5.5.5/server/solr-webapp/webapp/WEB-INF/lib

cp /usr/local/software/solr-5.5.5/dist/solr-dataimporthandler-extras-5.5.5.jar /usr/local/software/solr-5.5.5/server/solr-webapp/webapp/WEB-INF/lib

在solr-5.5.5/server/solr-webapp/webapp/WEB-INF下新建一个classes的文件夹

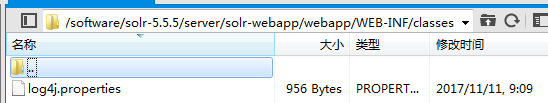


把solr-5.5.5/server/resources/log4j.properties复制到solr-5.5.5/server/solr-webapp/webapp/WEB-INF/classes下：

复制的命令是：

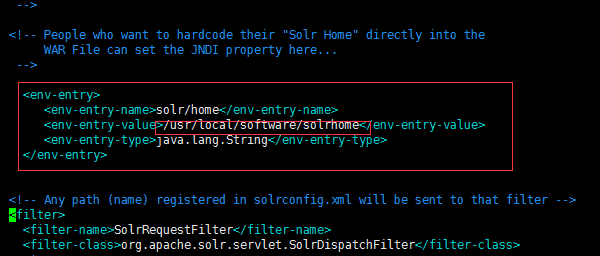
cp /usr/local/software/solr-5.5.5/server/resources/log4j.properties /usr/local/software/solr-5.5.5/server/solr-webapp/webapp/WEB-INF/classes





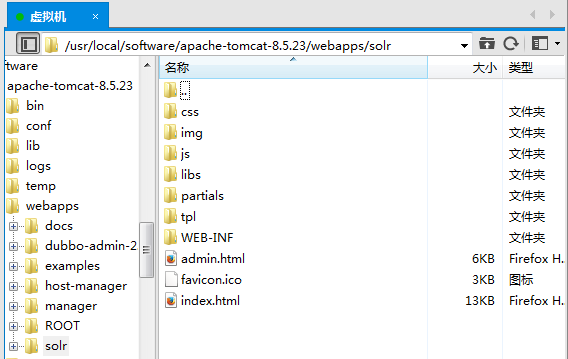
编辑solr-5.5.5/server/solr-webapp/webapp/WEB-INF下的web.xml

把这段的注释去掉，配置solrhome的位置



在tomcat的webapps下新建solr文件夹

将solr-5.5.5/server/solr-webapp/webapp中所有的内容复制到tomcat的webapps/solr下，注意只要webapp下的内容，不要webapp文件夹



复制的命令是：

cp -rf /usr/local/software/solr-5.5.5/server/solr-webapp/webapp/. /usr/local/software/apache-tomcat-8.5.23/webapps/solr



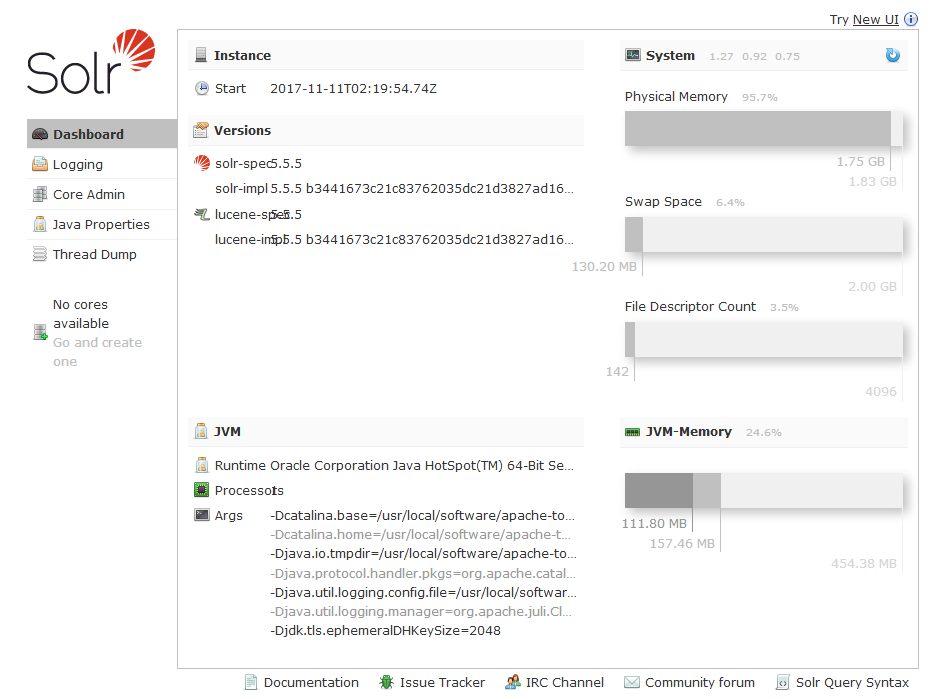
启动tomcat

访问

<http://www.vm.com:8080/solr/admin.html>

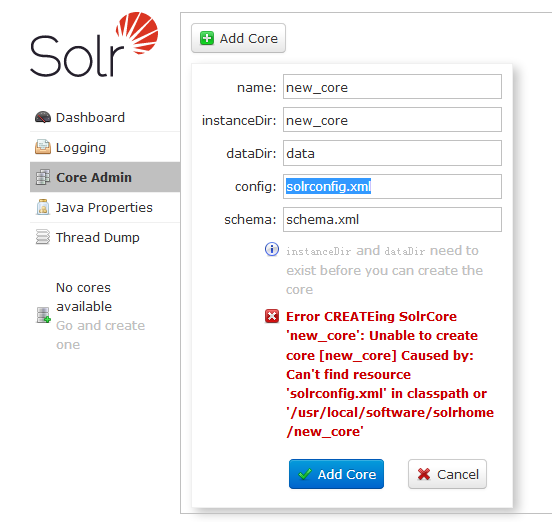
其中[www.vm.com:8080](http://www.vm.com:8080)替换成你虚拟机的地址和tomcat端口号

可以看见solr控制台，这个启动的过程比较慢，可能要几分钟

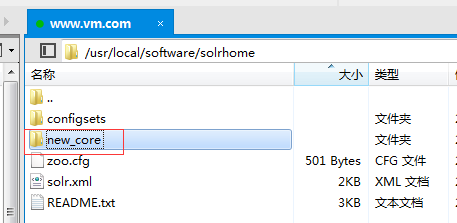


## 创建solr core

控制台创建core失败：



进入你的solrhome目录

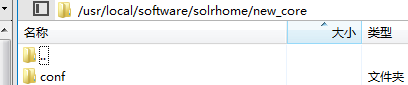


创建一个新的文件夹new\_core

将solrhome下的configsets/sample\_techproducts\_configs中的conf文件夹复制到new\_core中

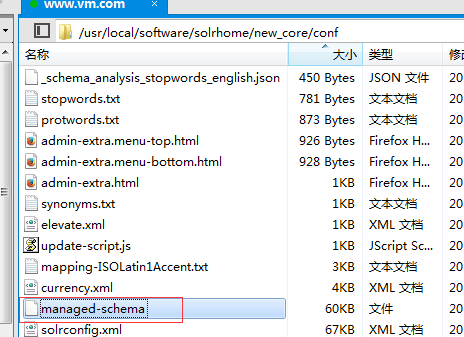
复制的命令是：

cp -r /usr/local/software/solrhome/configsets/sample\_techproducts\_configs/\* /usr/local/software/solrhome/new\_core



再去控制台创建core就可以成功了

## 配置中文分词



把new\_core/conf下的manage-schema重命名为schema.xml

编辑schema.xml，在文件末尾追加如下内容:

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

<analyzer class="org.wltea.analyzer.lucene.IKAnalyzer"/>

</fieldType>

<field name="item\_title" type="text\_ik" indexed="true" stored="true"/>

<field name="item\_price" type="long" indexed="true" stored="true"/>

<field name="item\_city\_id" type="long" indexed="true" stored="true"/>

<field name="item\_city\_name" type="string" indexed="true" stored="true" />

<field name="item\_image" type="string" indexed="true" stored="true" />

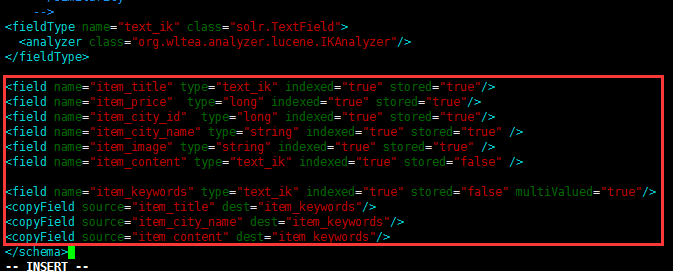
<field name="item\_content" type="text\_ik" indexed="true" stored="false" />

<field name="item\_keywords" type="text\_ik" indexed="true" stored="false" multiValued="true"/>

<copyField source="item\_title" dest="item\_keywords"/>

<copyField source="item\_city\_name" dest="item\_keywords"/>

<copyField source="item\_content" dest="item\_keywords"/>



参考资料中有中文分词的jar:



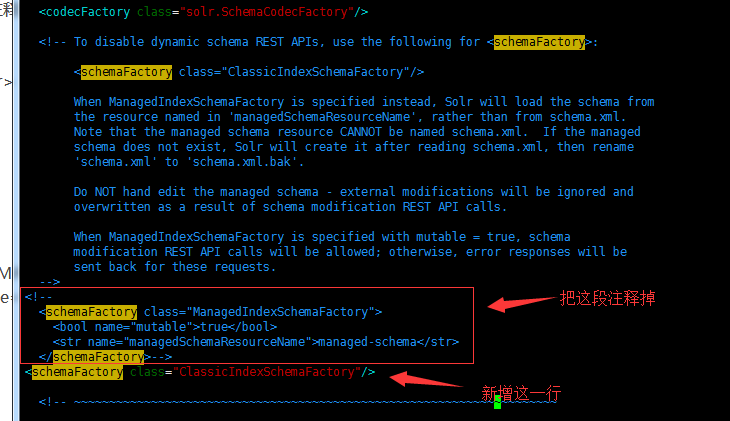
将分词的jar包上传到tomcat/webapps/solr/WEB-INF/lib下

编辑new\_core/conf/solrconfig.xml

将以下代码注释掉：

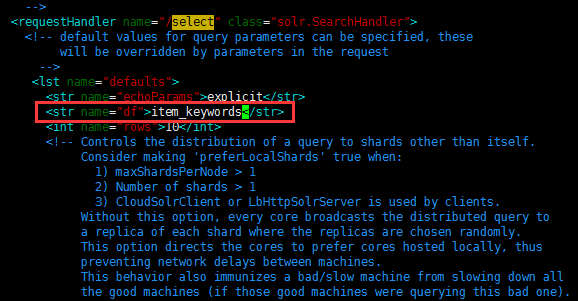
<schemaFactory class="ManagedIndexSchemaFactory">  
  <bool name="mutable">true</bool>  
 <str name="managedSchemaResourceName">managed-schema</str>  
</schemaFactory>

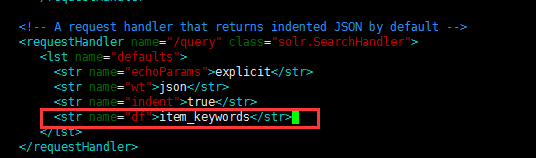
同时新增<schemaFactory class="ClassicIndexSchemaFactory"/>



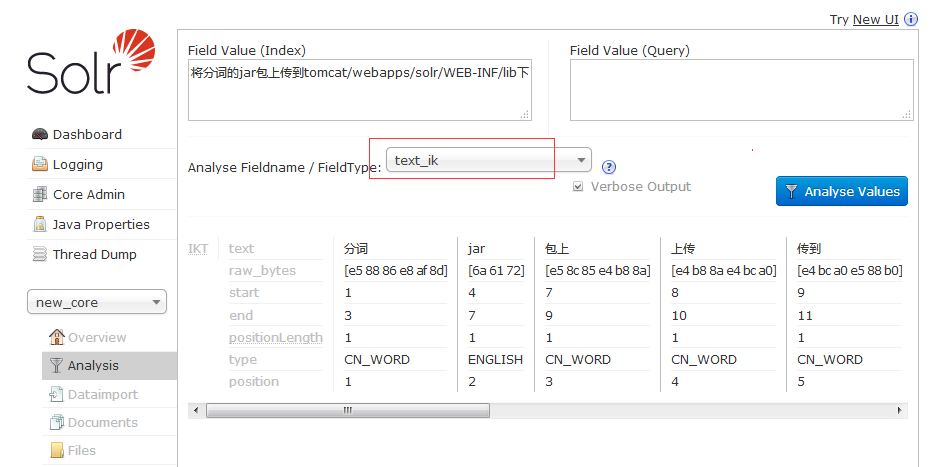
配置查询的默认字段：

在solrhome/new\_core/conf/solrconfig.xml中添加配置：



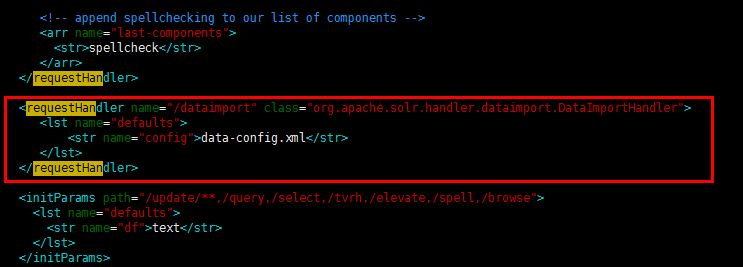


重启tomcat后可以使用分词器了。



## 配置数据库(选学)

在solrhome/new\_core/conf/solrconfig.xml中添加配置：



<requestHandler name="/dataimport" class="org.apache.solr.handler.dataimport.DataImportHandler">

<lst name="defaults">

<str name="config">data-config.xml</str>

</lst>

</requestHandler>

在同一目录下创建data-config.xml文件

<dataConfig>

<dataSource type="JdbcDataSource" driver="com.mysql.jdbc.Driver" url="jdbc:mysql://localhost:3306/travel\_db?characterEncoding=utf-8" user="root" password="root" batchSize="-1"/>

<document>

<entity name="hotel" query="select ID,TITLE,PRICE,IMAGE,CITY\_NAME, CITY\_ID from HOTEL" dataSource="JdbcDataSource">

<field column="ID" name="id" />

<field column="TITLE" name="item\_title" />

<field column="PRICE" name="item\_price" />

<field column="CONTENT" name="item\_content" />

<field column="IMAGE" name="item\_image" />

<field column="CITY\_NAME" name="item\_city\_name" />

<field column="CITY\_ID" name="item\_city\_id" />

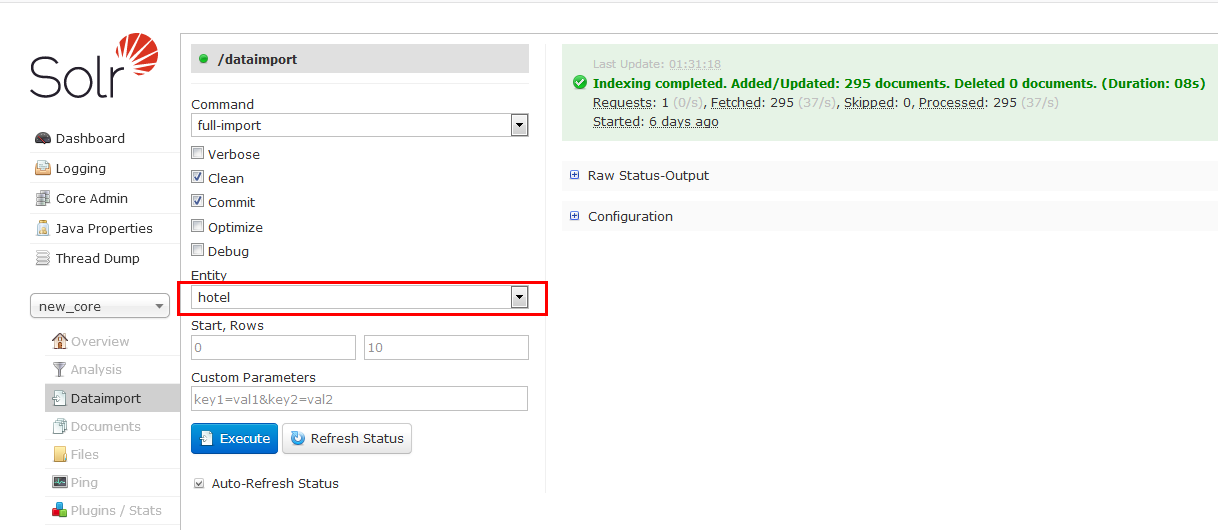
</entity>

</document>

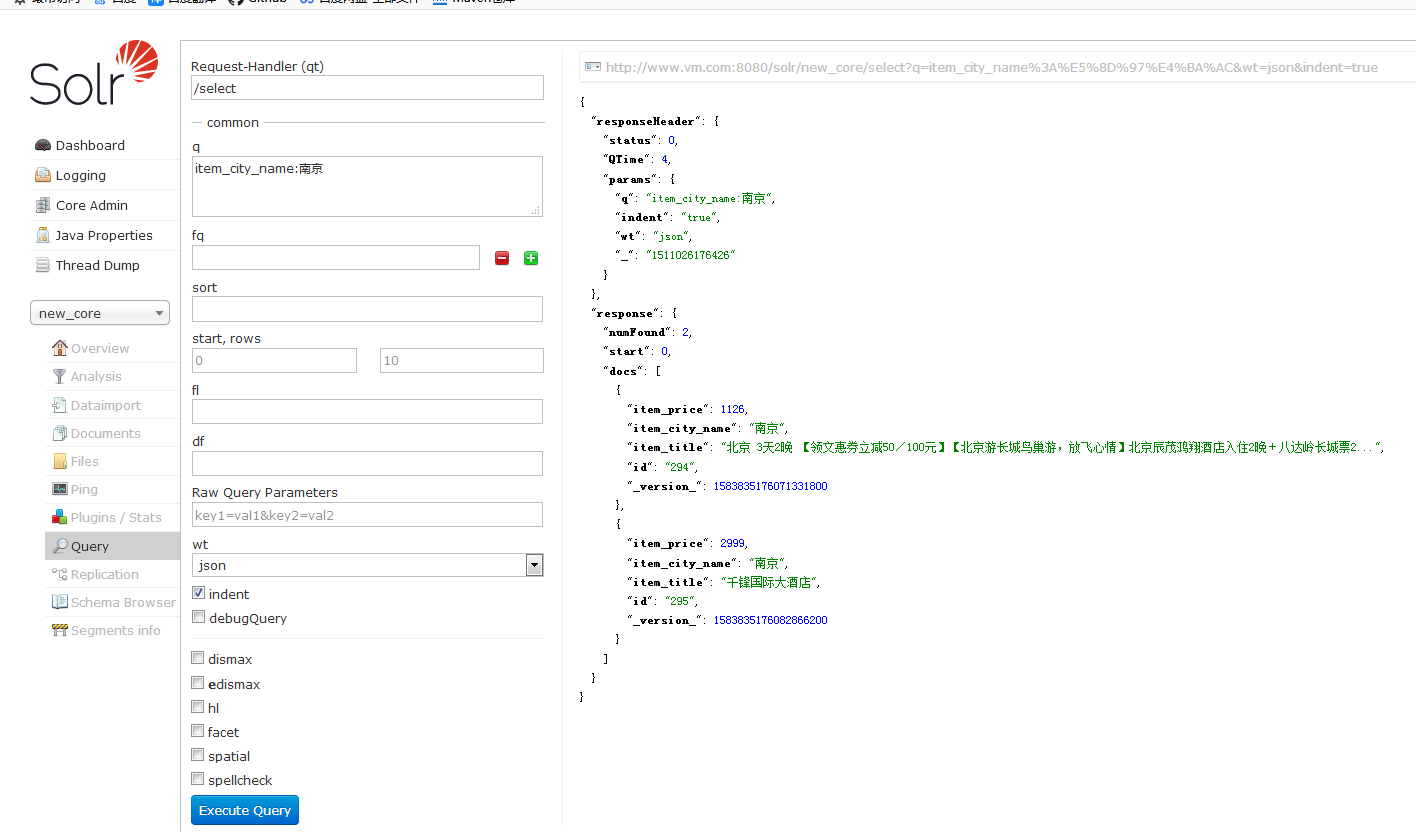
</dataConfig>

需要将mysql的jar包上传到tomcat/webapps/solr/WEB-INF/lib下

重启tomcat



q的格式是field:关键字。如果是\*:\*代表全查



# solr集群

Solr集群的系统架构



物理结构

三个Solr实例（ 每个实例包括两个Core），组成一个SolrCloud。

逻辑结构

索引集合包括两个Shard（shard1和shard2），shard1和shard2分别由三个Core组成，其中一个Leader两个Replication，Leader是由zookeeper选举产生，zookeeper控制每个shard上三个Core的索引数据一致，解决高可用问题。

用户发起索引请求分别从shard1和shard2上获取，解决高并发问题。

collection

Collection在SolrCloud集群中是一个逻辑意义上的完整的索引结构。它常常被划分为一个或多个Shard（分片），它们使用相同的配置信息。

比如：针对商品信息搜索可以创建一个collection。

collection=shard1+shard2+....+shardX

Core

每个Core是Solr中一个独立运行单位，提供 索引和搜索服务。一个shard需要由一个Core或多个Core组成。由于collection由多个shard组成所以collection一般由多个core组成。

Master或Slave

Master是master-slave结构中的主结点（通常说主服务器），Slave是master-slave结构中的从结点（通常说从服务器或备服务器）。同一个Shard下master和slave存储的数据是一致的，这是为了达到高可用目的。

Shard

Collection的逻辑分片。每个Shard被化成一个或者多个replication，通过选举确定哪个是Leader。

需要实现的solr集群架构



Zookeeper作为集群的管理工具。

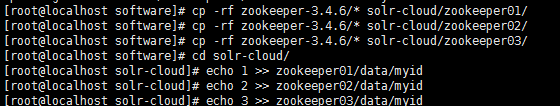
1. 集群管理：容错、负载均衡。
2. 配置文件的集中管理
3. 集群的入口

需要实现zookeeper 高可用。需要搭建集群。建议是奇数节点。需要三个zookeeper服务器。

在/usr/local/software/下新建solr-cloud文件夹，所有solr集群相关的配置都放在这个文件夹下。本教程是在一台机器上搭建的伪集群。如果搭建真正的集群，就把安装包上传到不同的服务器上，端口和ip及目录使用当前服务器的真实值即可。

## 安装zookeeper集群

复制三份单机版的zookeeper，分别复制到solr-cloud下的zookeeper01、zookeeper02、zookeeper03中。也可以使用zookeeper的安装包解压过来。



修改zookeeper0x/conf/zoo.conf，其中x代表1/2/3



dataDir分别是zookeeper01、zookeeper02、zookeeper03下的data目录，如果你是解压过来的，这个data需要你自己创建。如果你是复制的单机版，data下面可能有东西，把data文件夹下所有的文件夹都删掉，保证data文件夹是空的。

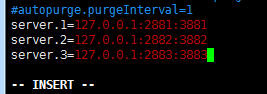
clientPort默认是2181，因为2181 被单机版的zookeeper占用了，所以这里换2281，zookeeper02就是2282，zookeeper03就是2283

三个zookeeper的zoo.conf文件末尾都要追加：

server.1=127.0.0.1:2881:3881

server.2=127.0.0.1:2882:3882

server.3=127.0.0.1:2883:3883



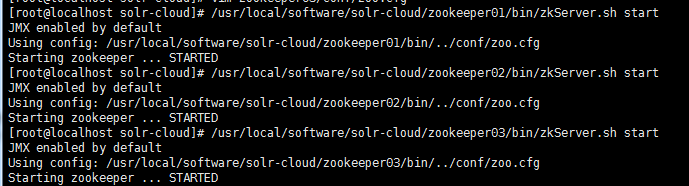
其中127.0.0.1应该改成你服务器的真实ip，因为我们搭在一台机器上，可以使用本机地址，避免你虚拟机ip改变后无法使用集群。

启动服务，命令是：

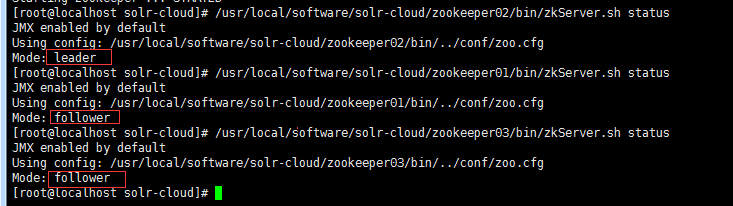
/usr/local/software/solr-cloud/zookeeper01/bin/zkServer.sh start

/usr/local/software/solr-cloud/zookeeper02/bin/zkServer.sh start

/usr/local/software/solr-cloud/zookeeper03/bin/zkServer.sh start

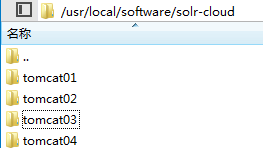


查看集群状态



## 安装tomcat

在solr-cloud下解压四份tomcat



修改每一个tomcat下的conf/server.xml，把里面的端口号都改掉，第二位都加1(因为是伪集群，一台机器上端口不能冲突。如果是真正的集群，不同的服务器上可以不改端口了)

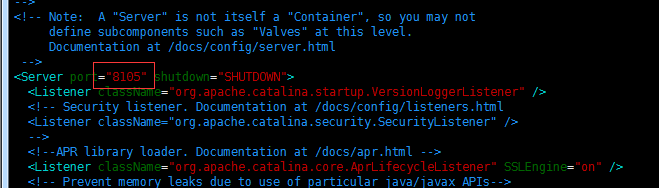
这几个端口默认的是8005,8080,8009

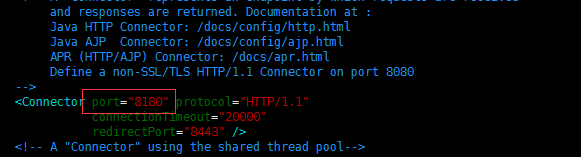
tomcat01分别改成：8105,8180,8109

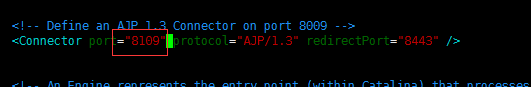
tomcat02分别改成：8205,8280,8209

tomcat03分别改成：8305,8380,8309

tomcat04分别改成：8405,8480,8409

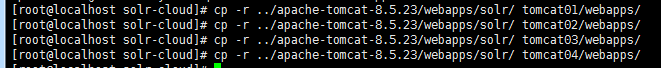




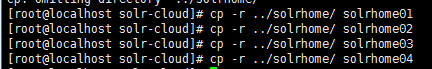


## 配置solr服务

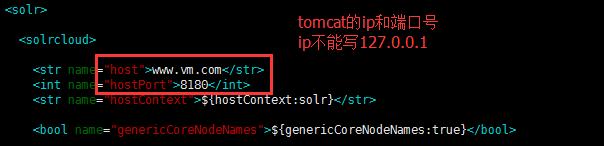
把单机版的solr复制四份过来或者分别上传solr.war到这几个tomcat的webapps下



把solrhome复制四份过来，或者按照1.1中的步骤创建四个新的solrhome



vim solrhome01/solr.xml，solrhome02、solrhome03、solrhome04下面的也要修改



host是你tomcat服务器的ip或域名，这里装在一台机器上，不要写127.0.0.1

solrhome01/solr.xml 的hostPort是8180，代表使用tomcat01

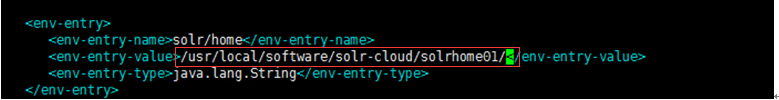
solrhome02/solr.xml 的hostPort是8280，代表使用tomcat02

solrhome03/solr.xml 的hostPort是8380，代表使用tomcat03

solrhome04/solr.xml 的hostPort是8480，代表使用tomcat04

修改每个tomcat的/webapps/solr/WEB-INF/web.xml

vim tomcat01/webapps/solr/WEB-INF/web.xml，tomcat02,tomcat03,tomcat04也要修改



tomcat01对应的是solrhome01

tomcat02对应的是solrhome02

tomcat03对应的是solrhome03

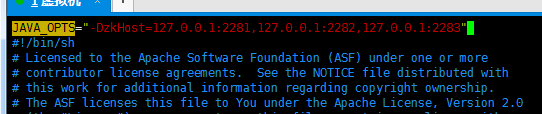
tomcat04对应的是solrhome04

修改tomcat的bin/catalina.sh，四个tomcat都要修改

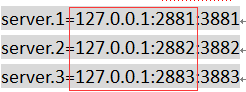
vim tomcat01/bin/catalina.sh

在文件第一行加上如下配置，此处的ip应该是你服务器的外网ip：

JAVA\_OPTS="-DzkHost=127.0.0.1:2281,127.0.0.1:2282,127.0.0.1:2283"



其中这三个地址，对应的是2.1中zookeeper配置文件中的地址，就是你zookeeper节点的ip和端口号，如图：



## 上传solr配置文件到zookeeper

每个solrhome下都有自己的配置文件，需要上传一份到zookeeper，让zookeeper统一管理。

使用工具上传配置文件：上传solrhome1-4中任意一个就行。使用的是solr安装包下的工具

cd /usr/local/software/solr-5.5.5/server/scripts/cloud-scripts

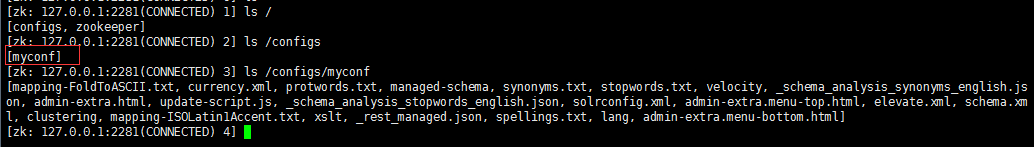
./zkcli.sh -zkhost 127.0.0.1:2281,127.0.0.1:2282,127.0.0.1:2283 -cmd upconfig -confdir /usr/local/software/solr-cloud/solrhome01/new\_core/conf -confname myconf

如果在2.3中不是复制的单机版solrhome，而是新建的solrhome，是没有new\_core这个核心文件的，请参照1.3和1.4中的方法，创建一个new\_core再上传。

连接zookeeper查看是否上传成功：

cd /usr/local/software/solr-cloud/zookeeper01/bin

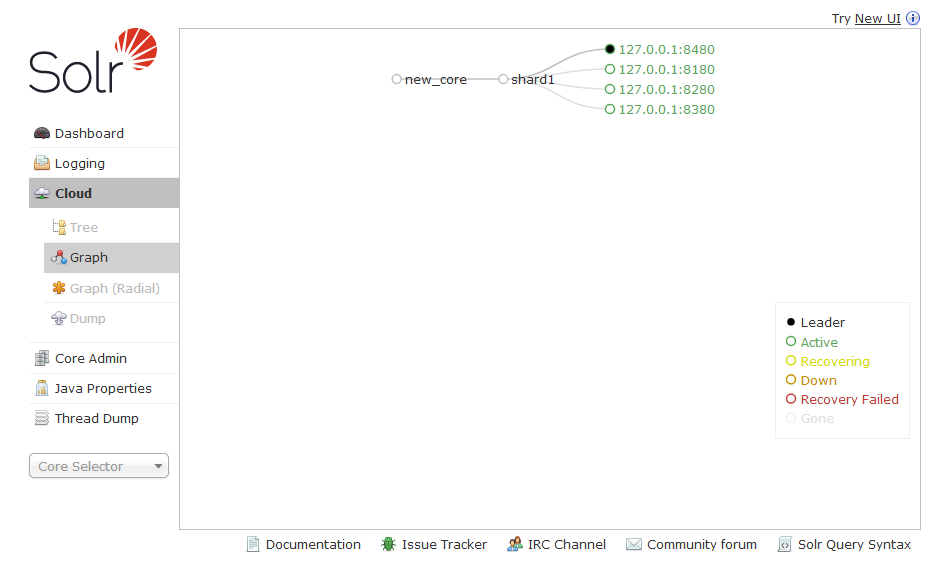
./zkCli.sh -server 127.0.0.1:2281



quit退出

## 启动集群

启动每一台tomcat。启动的时间很长，需要等几分钟

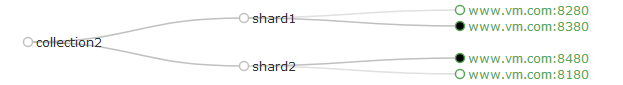
访问任意一个tomcat，可以在cloud中看到集群信息。注意core不要随便删，集群可能会崩掉，自己在测试机上多测几回再去生产环境弄。

增加solr core并分片的命令。

http://www.vm.com:8180/solr/admin/collections?action=CREATE&name=collection2&numShards=2&replicationFactor=2



再去查看管理后台：



删除solr core（慎用，操作不好的话集群有可能崩掉，还要重新搭建）

<http://www.vm.com:8180/solr/admin/collections?action=DELETE&name=new_core>



# java操作solr

首先在项目中引用solr的jar包solrj

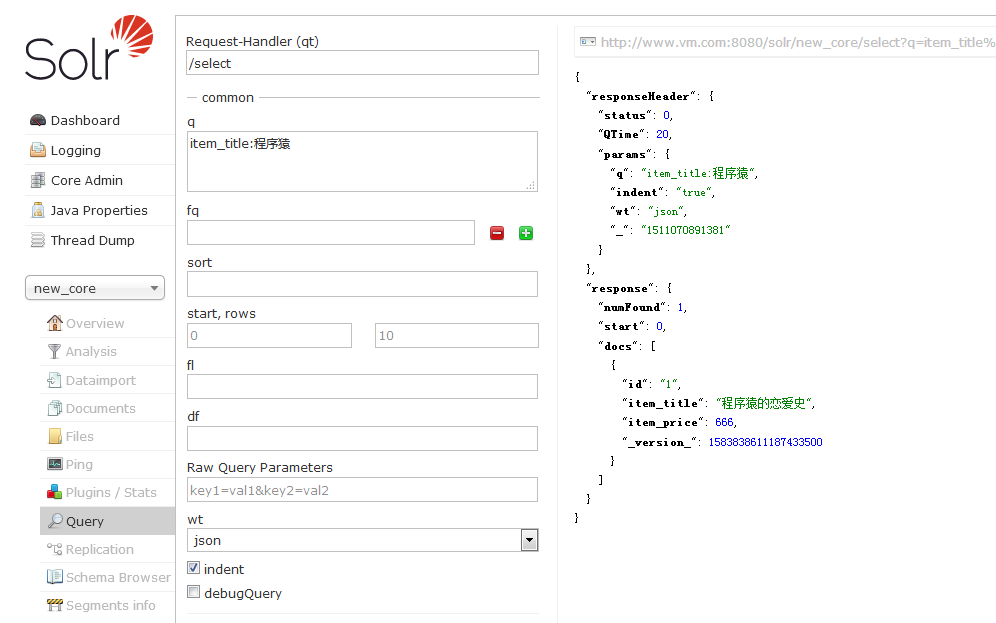
<!--solr-->  
<dependency>  
 <groupId>org.apache.solr</groupId>  
 <artifactId>solr-solrj</artifactId>  
 <version>5.5.5</version>  
</dependency>

## 单机版solr

### 添加索引

**@Test  
public void** testCreate() **throws Exception {  
 //链接到solr的new\_core核心文件  
 String url = "http://www.vm.com:8080/solr/new\_core"**;  
 **HttpSolrClient server = new** HttpSolrClient(**url**);  
 **//创建文档  
 SolrInputDocument doc = new** SolrInputDocument();  
 **//这个field是在solr core中配置过的  
 doc**.addField(**"id"**, **"1"**);  
 **doc**.addField(**"item\_title"**, **"程序猿的恋爱史"**);  
 **doc**.addField(**"item\_content"**, **"别扯了，程序猿怎么会有女朋友"**);  
 **doc**.addField(**"item\_price"**, **666**);  
 **server**.add(**doc**);  
 **server**.commit();  
 **server**.close();  
 **System**.***out***.println(**"创建成功！"**);  
**}**

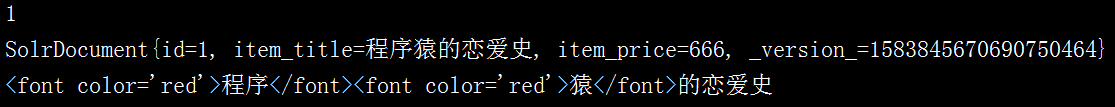
去控制台们可以看到新添加的索引



### 查询索引

**@Test  
public void** testQuery() **throws Exception {  
 String url = "http://www.vm.com:8080/solr/new\_core"**;  
 **HttpSolrClient server = new** HttpSolrClient(**url**);  
 **//定义查询内容  
 SolrQuery query = new** SolrQuery(**"item\_title:程序猿"**); **//定义查询内容  
 query**.setStart(**0**);**//起始页  
 query**.setRows(**3**);**//每页显示数量  
 query**.setSort(**"id"**, **SolrQuery**.ORDER.***asc***);**//排序  
 query**.setHighlight(**true**);**//设置高亮  
 query**.addHighlightField(**"item\_title"**);**//设置高亮字段  
 // 设置高亮的样式  
 query**.setHighlightSimplePre(**"<font color='red'>"**);  
 **query**.setHighlightSimplePost(**"</font>"**);  
 **QueryResponse rsp = server**.query(**query**);  
 **SolrDocumentList results = rsp**.getResults();  
 **System**.***out***.println(**results**.getNumFound());**//查询总条数  
 //取高亮,id是key** Map**<String**, Map**<String**, List**<String>>> highLightings = rsp**.getHighlighting();  
 **for** (**SolrDocument doc : results**) **{  
 System**.***out***.println(**doc**);  
 **//获取当前对象的高亮** List**<String> list = highLightings**.**get**(**doc**.get(**"id"**)).**get**(**"item\_title"**);  
 **String itemTitle = null**;  
 **if** (**list != null && list**.**size**() **> 0**) **{  
 itemTitle = list**.**get**(**0**);  
 **} else {  
 itemTitle =** (**String**) **doc**.get(**"item\_title"**);  
 **}  
 System**.***out***.println(**itemTitle**);  
 **System**.***out***.println(**"------------------------------"**);  
 **}  
 server**.close();  
**}**

看控制台的查询结果：



### 删除索引

**@Test  
public void** testDelete() **throws Exception {  
 String url = "http://www.vm.com:8080/solr/new\_core"**;  
 **HttpSolrClient server = new** HttpSolrClient(**url**);  
 **//删除所有分词，基于查询删除  
 //server.deleteByQuery("\*:\*");  
 //根据id删除所有分词  
 server**.deleteById(**"1"**);  
 **server**.commit();  
 **server**.close();  
 **System**.***out***.println(**"删除成功"**);  
**}**

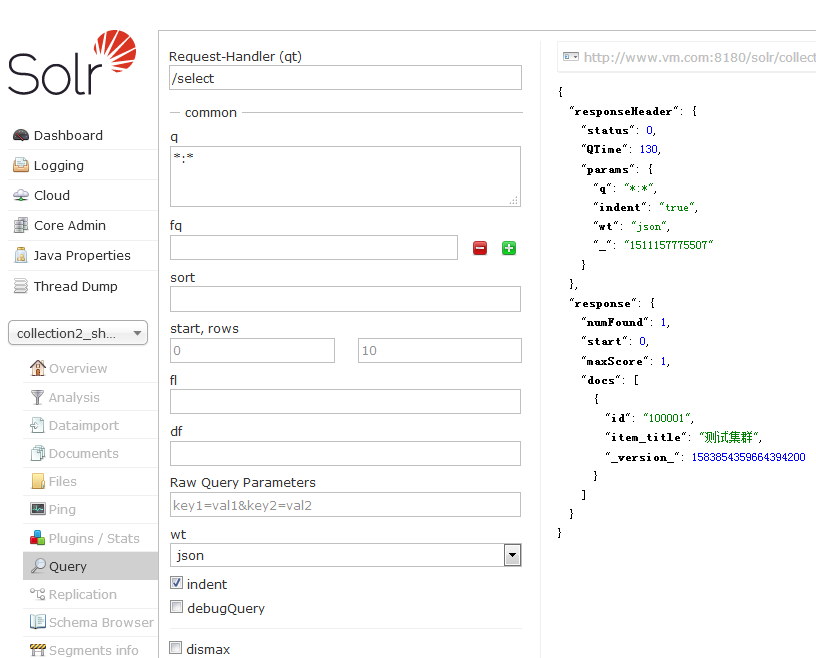
去管理后台查看，索引被删除：



## 集群版solr

### 添加索引

**// zookeeper地址  
private static String** zkHostString **= "www.vm.com:2281,www.vm.com:2282,www.vm.com:2283"**;  
**// solr core的名字默认名称，不加shard  
private static String** defaultCollection **= "collection2"**;  
  
**// solr集群实例  
private CloudSolrClient cloudSolrClient**;  
  
**// 测试方法之前构造 cloudSolrClient  
@Before  
public void** init() **{  
 cloudSolrClient = new** CloudSolrClient(zkHostString);  
 **cloudSolrClient**.setDefaultCollection(defaultCollection);  
 **cloudSolrClient**.setZkClientTimeout(**3000**);  
 **cloudSolrClient**.setZkConnectTimeout(**3000**);  
 **cloudSolrClient**.connect();  
**}  
  
@After  
public void** destroy() **throws Exception {  
 if** (**cloudSolrClient != null**) **{  
 cloudSolrClient**.close();  
 **}  
}  
  
// 在solrCloud上创建索引  
@Test  
public void** testCreateIndex() **throws SolrServerException**,  
 **IOException {  
 SolrInputDocument document = new** SolrInputDocument();  
 **document**.addField(**"id"**, **"100001"**);  
 **document**.addField(**"item\_title"**, **"测试集群"**);  
 **cloudSolrClient**.add(**document**);  
 **cloudSolrClient**.commit();  
 **System**.***out***.println(**"添加成功"**);  
**}**



### 查询索引

**// 搜索索引  
@Test  
public void** testSearchIndex() **throws Exception {  
 SolrQuery query = new** SolrQuery();  
 **query**.setQuery(**"\*:\*"**);  
 **QueryResponse response = cloudSolrClient**.query(**query**);  
 **SolrDocumentList docs = response**.getResults();  
  
 **System**.***out***.println(**"文档个数：" + docs**.getNumFound());  
 **System**.***out***.println(**"查询时间：" + response**.getQTime());  
  
 **for** (**SolrDocument doc : docs**) **{  
 String title =** (**String**)**doc**.getFieldValue(**"item\_title"**);  
 **String id =** (**String**) **doc**.getFieldValue(**"id"**);  
 **System**.***out***.println(**"id: " + id**);  
 **System**.***out***.println(**"item\_title: " + title**);  
 **}  
}**

### 删除索引

**// 删除索引  
@Test  
public void** testDeleteIndexFromSolrCloud() **throws SolrServerException**, **IOException {  
 // 根据id删除  
 cloudSolrClient**.deleteById(**"100001"**);  
 **// 根据多个id删除  
 // cloudSolrClient.deleteById(ids);  
 // 查询条件删除  
 // cloudSolrClient.deleteByQuery("item\_title:测试");  
 // 提交  
 cloudSolrClient**.commit();  
**}**

去管理后台查看，索引被删除：

