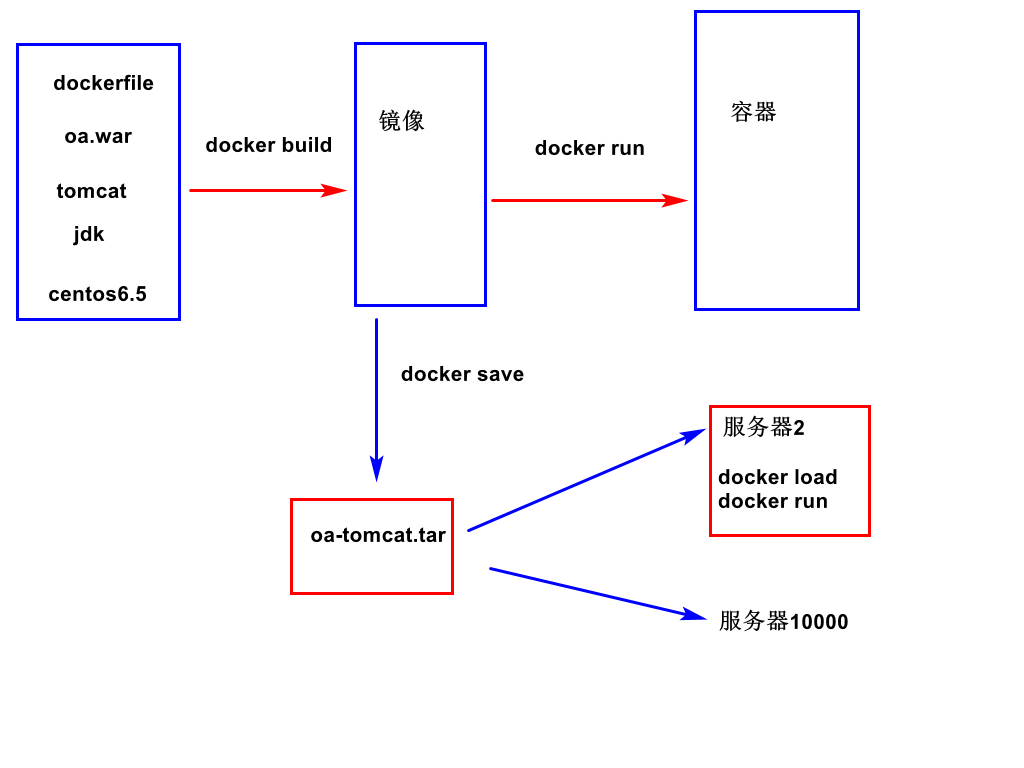
# 简历

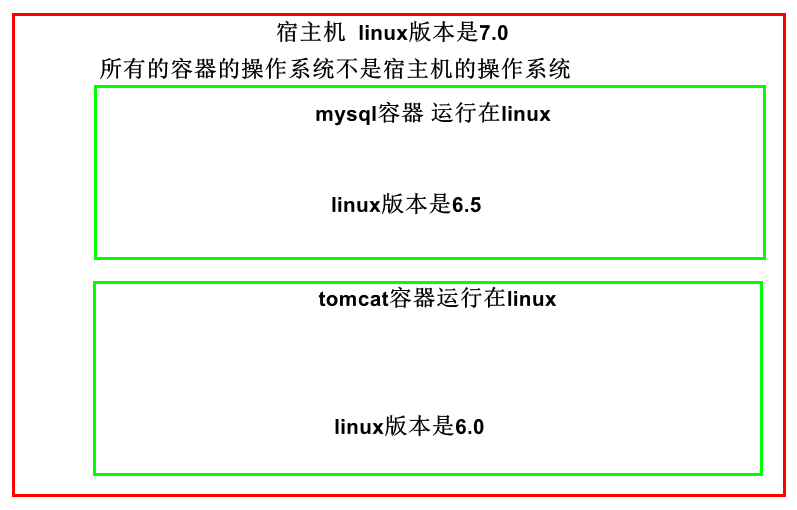
各个中心同学把简历发给班长，班长压成一个rar，[发到我的邮箱474530502@qq.com](mailto:发到我的邮箱474530502@qq.com),周四看

# 复习



# Docker

## Dockerfile制作镜像



Dockerfile是docker应用的核心，通过dockerfile能制作出新的镜像，如图- 150所示。

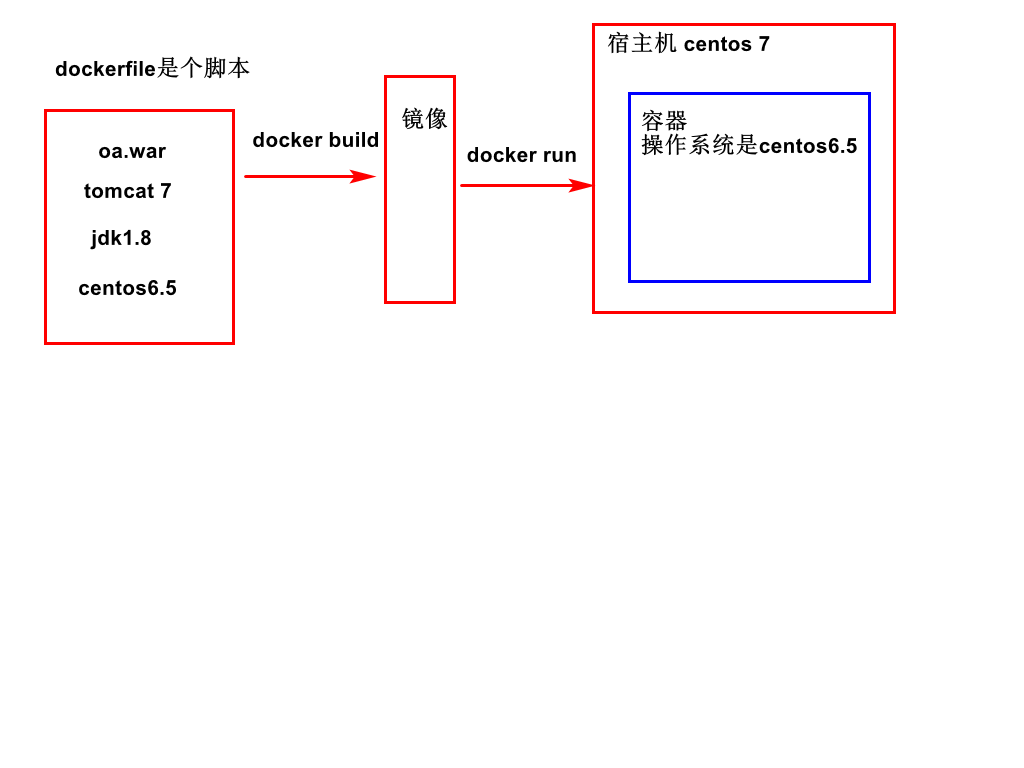
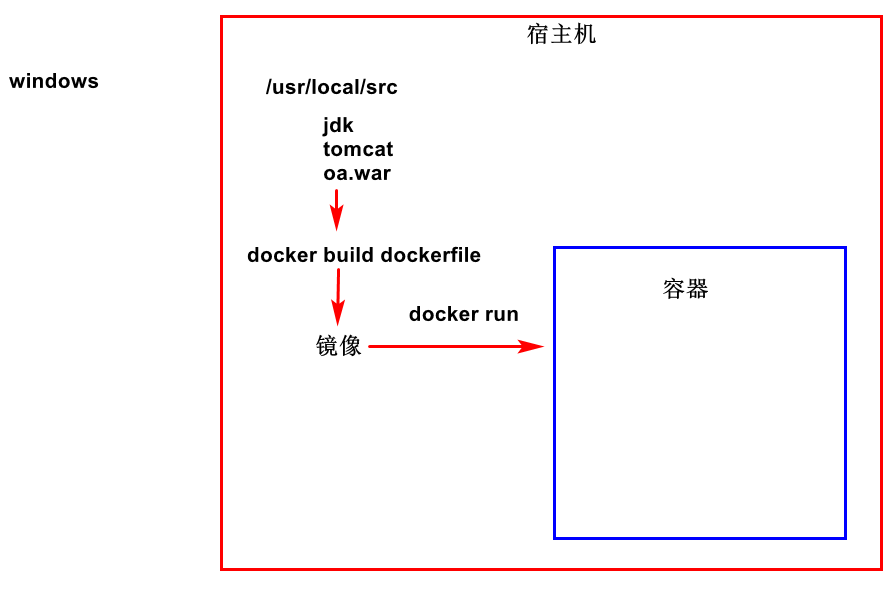


图- 150



### 关键字及含义

|  |  |  |
| --- | --- | --- |
| 序号 | 关键字 | 说明 |
|  | FROM | 指定基础镜像的来源 |
|  | MAINTAINER | 作者 |
|  | ADD | 复制文件，会自动解压 |
|  | WORKDIR | 设置当前工作目录 cd |
|  | VOLUME | 设置数据卷，挂载主机目录 |
|  | EXPOSE | 指定对外暴漏的端口 |
|  | RUN | 执行命令 sh |
|  | CMD | 执行命令 exec，一个Dockerfile只能一个 |
|  | COPY | 复制文件 |
|  | ENTRYPOINT | docker run时参数可以覆盖，指定参数值 |

### 查看镜像

//查看装好的centos镜像

docker images

命令执行完后结果如图-132 所示。

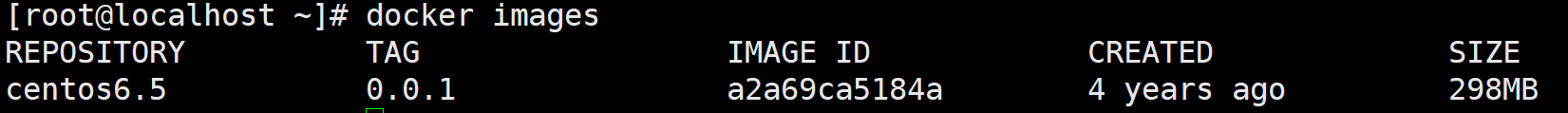
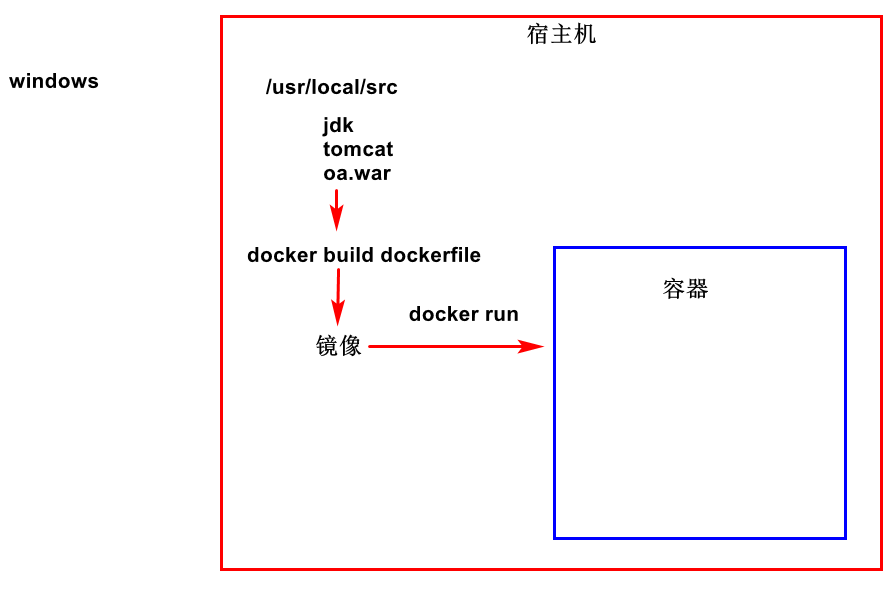


图- 132

### 生成镜像



1. 上传“\培优课前资料\亿发,移动端课前资料\docker”文件夹中的jdk-8u51-linux-x64.tar.gz到虚拟机的\usr\local\src。
2. 上传\亿发,移动端课前资料\docker \apache-tomcat-7.0.55.tar.gz到虚拟机的\usr\local\src
3. 使用eclipse创建出一个maven webapp项目,项目名为oa,修改index.jsp中的内容，选中项目右键 run as 🡪maven install打包出war,到target文件夹中找到oa.war。
4. 上传oa.war到虚拟机的\usr\local\src
5. 上传\亿发,移动端课前资料\docker \Dockerfile到虚拟机的\usr\local\src

Dockerfile文件中内容如下：

#添加contos6.5 image

FROM a2a69ca5184a

#添加JDK1.8 centos+jdk src是容器的

ADD jdk-8u51-linux-x64.tar.gz /usr/local/src

#java\_home是容器的

ENV JAVA\_HOME=/usr/local/src/jdk1.8.0\_51

ENV PATH=$JAVA\_HOME/bin:$PATH

ENV CLASSPATH=.:$JAVA\_HOME/lib/dt.jar:$JAVA\_HOME/lib/tools.jar

#centOS6.5+JDK1.8+tomcat7

ADD apache-tomcat-7.0.55.tar.gz /usr/local/src

ENV CATALINA\_HOME /usr/local/src/apache-tomcat-7.0.55

ENV PATH=$PATH:$CATALINA\_HOME/bin

#添加oa.war包文件

WORKDIR $CATALINA\_HOME/webapps

RUN mkdir oa

COPY oa.war $CATALINA\_HOME/webapps/oa/oa.war

WORKDIR $CATALINA\_HOME/webapps/oa

RUN jar xvf oa.war

RUN rm oa.war

#对外暴露的端口号

EXPOSE 8080

CMD ["/usr/local/src/apache-tomcat-7.0.55/bin/catalina.sh","run"]

1. 制作新的镜像文件

镜像名称是oa-tomcat,版本是0.0.1

cd /usr/local/src

// ./代表当前目录

docker build -t oa-tomcat:0.0.1 -f Dockerfile ./

制作成功会显示“Successfully tagged oa-tomcat:0.0.1”

1. docker images 查看镜像，如图- 151所示



图- 151

## 使用自己制作的镜像创建容器

1.创建容器

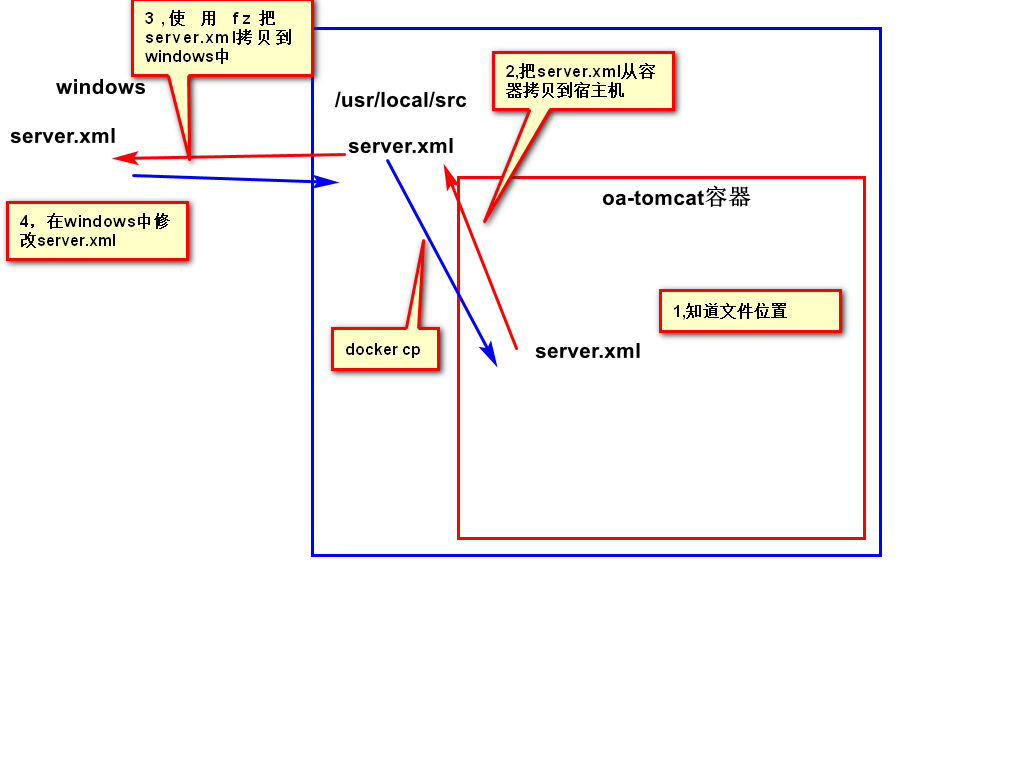
docker run -d -p 9080:8080 --net dockernet --ip 172.18.0.13 --name oa-tomcat oa-tomcat:0.0.1

2.访问tomcat和oa

访问<http://虚拟机ip:9080>能看到tomcat首页

访问http://虚拟机ip:9080/oa/index.jsp能看到oa首页

1. 查看容器内容文件结构



[root@localhost src]# docker ps -a

[root@localhost src]# docker exec -it oa-tomcat bash

bash-4.1# cd /usr/local/src

bash-4.1# ls

apache-tomcat-7.0.55 jdk1.8.0\_51

bash-4.1# cd apache-tomcat-7.0.55/

bash-4.1# ls

LICENSE RELEASE-NOTES bin lib temp work

NOTICE RUNNING.txt conf logs webapps

bash-4.1# cd webapps

bash-4.1# ls

ROOT docs examples host-manager manager oa

bash-4.1# cd oa

bash-4.1# ls

META-INF WEB-INF index.jsp

bash-4.1# cd ../

bash-4.1# cd ../

bash-4.1# pwd

/usr/local/src/apache-tomcat-7.0.55

bash-4.1# cd conf

bash-4.1# ls

server.xml

bash-4.1# exit

1. 把容器中的文件拷贝出来

[root@localhost src]# pwd

/usr/local/src

//.代表当前目录

[root@localhost src]# docker cp oa-tomcat:/usr/local/src/apache-tomcat-7.0.55/conf/server.xml .

[root@localhost src]# ls

server.xml

1. 修改server.xml

把server.xml拷贝到windows中。

在124行host标签中添加context标签

<Host name="localhost" appBase="webapps"

unpackWARs="true" autoDeploy="true">

<Context path="" docBase="/usr/local/src/apache-tomcat-7.0.55/webapps/oa" debug="0" reloadable="true" crossContext="true"/>

1. 拷贝server.xml到容器中

把windows中的server.xml拷贝到虚拟机的/usr/local/src中。

[root@localhost src]# pwd

/usr/local/src

[root@localhost src]# docker cp server.xml oa-tomcat:/usr/local/src/apache-tomcat-7.0.55/conf/

1. 重启容器

[root@localhost src]# docker ps -a

[root@localhost src]# docker restart oa-tomcat

1. 浏览器中访问

<http://192.168.216.201:9080/index.jsp能看到oa>首页

## 生成镜像文件

生成镜像文件tar的作用如图-152 所示.

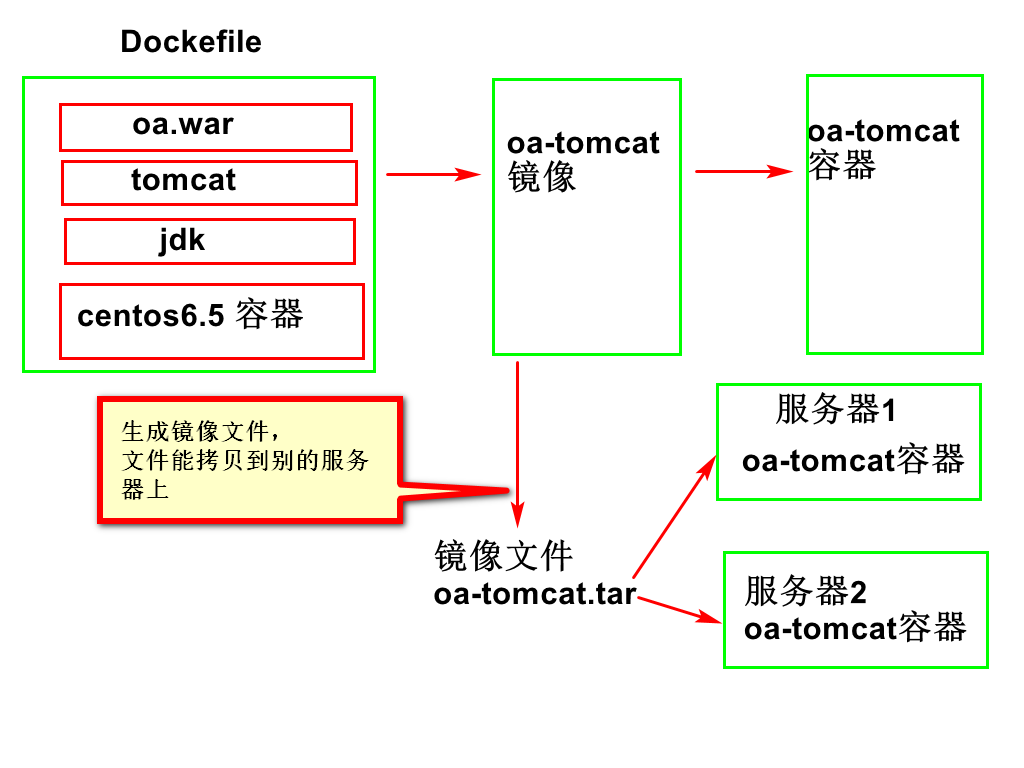


图- 152

[root@localhost src]# docker images

[root@localhost src]# docker save -o oa-tomcat-0.0.1.tar oa-tomcat:0.0.1

[root@localhost src]# ls

oa-tomcat-0.0.1.tar

## 使用制作的镜像文件创建容器

[root@localhost ~]# docker ps -a

[root@localhost ~]# docker stop oa-tomcat

//移出容器

[root@localhost ~]# docker rm oa-tomcat

//移出镜像

[root@localhost ~]# docker rmi oa-tomcat:0.0.1

Untagged: oa-tomcat:0.0.1

Deleted: sha256:1274a0931b333171acc17751da76407b935999ad6ee0cb6078c503828c15a3e6

//没有oa-tomcat镜像

[root@localhost src]# docker images

[root@localhost ~]# cd /usr/local/src

[root@localhost src]# ls

oa-tomcat-0.0.1.tar

[root@localhost src]# docker load -i oa-tomcat-0.0.1.tar

[root@localhost src]# docker images

REPOSITORY TAG IMAGE ID CREATED SIZE

oa-tomcat 0.0.1 1274a0931b33 15 hours ago 653MB

[root@localhost src]# docker run -d -p 9180:8080 --net dockernet --ip 172.18.0.19 --name oa-tomcat2 oa-tomcat:0.0.1

b617901309f6db4e4e49a60be22f8e929b1b3b4d0c24e3a55588ff61224fbe61

[root@localhost src]# docker ps -a

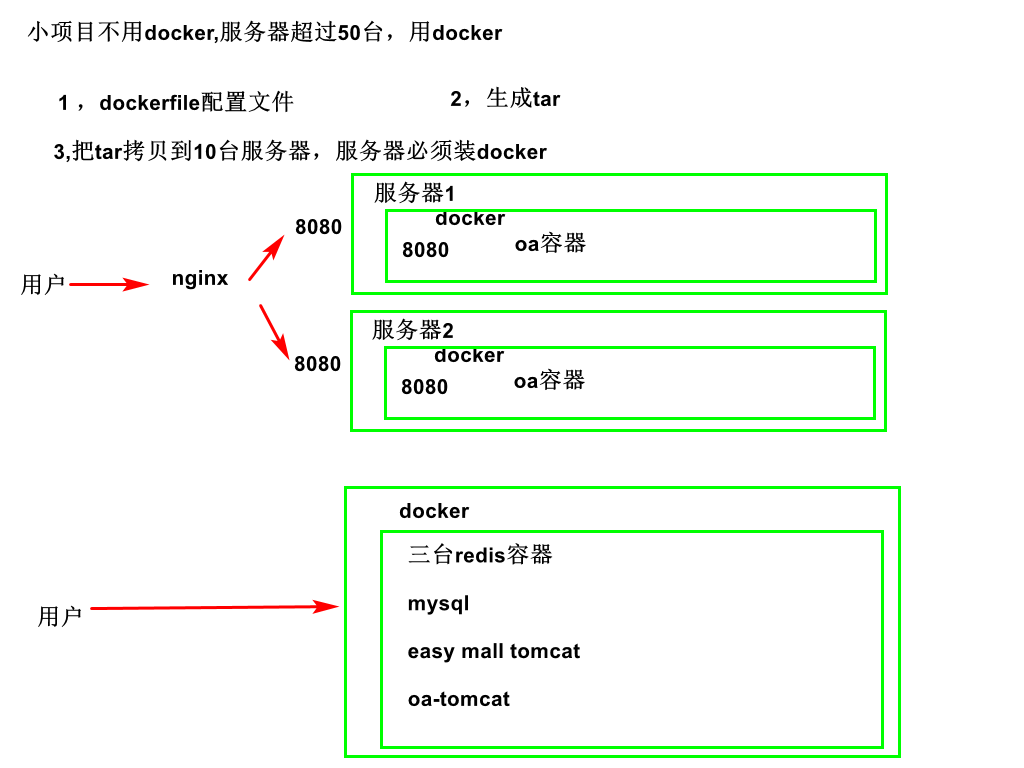
浏览器访问

[http://192.168.216.201:9180](http://192.168.216.201:9180/oa/index.jsp)

http://192.168.216.201:9180/oa/index.jsp

## 总结和拓展

Docker容器一般用于部署重复相同的服务，如：redis集群，tomcat集群等。适用于快速高效集群扩展。整个过程全自动，省去了运维成本。



有多台服务器装同样的环境，使用docker优点是装的速度快，环境一致

在一台服务器用docker装出redis容器，nginx容器，mysql容器,tomcat容器，优点是可限制每个容器使用的cpu,内存，磁盘，

不用docker,在一台服务器上装nginx,redis,mysql,tomcat,有一个服务出问题，别的服务也会出问题。

小的项目不用docker

## 重装docker

课上不用练习重装docker。

### 卸载老版本的 docker 及其相关依赖

sudo yum remove docker docker-common container-selinux docker-selinux docker-engine

### 安装 yum-utils，它提供了 yum-config-manager，可用来管理yum源

sudo yum install -y yum-utils

### 添加yum源

sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo

### 更新索引

sudo yum makecache fast

### 安装 docker-ce

sudo yum install docker-ce

# 全文检索

## 概念介绍

### 分词

用mysql也可以检索，用的关键字 where title like ‘%联想%’

百度 抓网页，网页有很多字符 java编程思想，进行中文分词 分成java /编程/思想

分词是把一串文字分割成多个关键词，如把java编程思想分割成java,编程，思想三个关键词，关键词也叫索引词。可以根据字典和统计进行分词

找到”\亿发,移动端课前资料\solr\solrLinux安装包\02-所需配置文件\solr配置jar包\ik-analyzer-5.3.0.jar”。

把jar改名成rar,用winrar解压jar，找到\org\wltea\analyzer\dic文件夹中。main2012.dic中存放了27万个词。

中文分词

字典[java,编程,语言,北京天安门]

网页中有字符串“java是编程语言”

字典中关键词最大长度是5

候选词：“java是” 不在字典中，去掉最右边的一个字符

候选词:“java” java在字典中，得到到关键词[java]

候选词: 是编程语言

候选词: 是编程语

候选词: 是编程

候选词: 是编

候选词: 是 单字是关键词，得到到关键词[java,是]

代码如下

**public** **class** AppMain {

//字典

**public** **static** ArrayList<String> *dic*=

**new** ArrayList<>();

**public** **static** **void** main(String[] args) {

//向字典添加关键词

*dic*.add("java");

*dic*.add("编程");

*dic*.add("语言");

*dic*.add("北京天安门");

String text="java是编程语言";

List<String> result=*segment*(text);

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

}

//分词

**public** **static** List<String> segment(String text)

{

//放分词结果

ArrayList<String> result=**new** ArrayList<>();

//while()分多次取出候选词

**while**(text.length()>0){

//字典中关键词最大长度是5

**int** max\_length=5;

**int** len=max\_length;

**if** (text.length()<max\_length)

{

len=text.length();

}

String tryWord=text.substring(0,len);

//1,判断候选词在不在字典中

**while**(!*dic*.contains(tryWord)){

//2,判断是不是单字

**if** (tryWord.length()==1)

{

//是单字，退出while

**break**;

}

//3,从右边去掉一个字符

tryWord=tryWord.substring

(0, tryWord.length()-1);

}

result.add(tryWord);

//去掉最左边的关键词

text=text.substring(tryWord.length());

}

**return** result;

}

}

### 倒排索引

Mysql查询

每次查询都去判断表中每一行有没有关键词

第一行 Java是编程语言

第二行 Java工程师

普通索引是在文档中找词，如在“java是编程语言”中查找java 。

倒排索引是根据词来找文档。

文档表

|  |  |
| --- | --- |
| 位置 | 内容 |
| 1 | Java是编程语言 |
| 2 | Java编程思想 |

索引表

|  |  |
| --- | --- |
| 关键词 | 位置 |
| java | 1，2 |
| 语言 | 1 |

## Lucene

### Lucene说明

Lucene是apache软件基金会的一个项目，是一个开放源代码的全文检索引擎工具包，在Java开发环境里Lucene是一个成熟的免费开源工具。就其本身而言，Lucene是当前以及最近几年最受欢迎的免费Java信息检索程序库。

### 导入测试jar包

如果ik依赖出错，把我网盘中的org\wltea\analyzer \ ik-analyzer-2012FF\_u1.jar拷贝到你电脑中

<!-- 全文检索lucene -->

<dependency>

<groupId>org.apache.lucene</groupId>

<artifactId>lucene-core</artifactId>

<version>4.10.2</version>

</dependency>

<dependency>

<groupId>org.apache.lucene</groupId>

<artifactId>lucene-analyzers-common</artifactId>

<version>4.10.2</version>

</dependency>

<dependency>

<groupId>org.apache.lucene</groupId>

<artifactId>lucene-queryparser</artifactId>

<version>4.10.2</version>

</dependency>

<!-- IK分词器 -->

<dependency>

<groupId>org.wltea.analyzer</groupId>

<artifactId>ik-analyzer</artifactId>

<version>2012FF\_u1</version>

</dependency>

### 创建索引

lucene/LuceneTest

**public** **class** LuceneTest {

@Test

**public** **void** createIndex() **throws** Exception

{

//1,指定文件夹保存数据

File file=**new** File("./index");

Directory directory=FSDirectory.*open*(file);

//2,指定分词器

Analyzer analyzer=**new** IKAnalyzer();

//3,设置配置信息

IndexWriterConfig config=**new** IndexWriterConfig

(Version.***LUCENE\_4\_10\_2***, analyzer);

//4,创建写数据的对象

IndexWriter indexWriter=

**new** IndexWriter(directory, config);

//5,创建document

Document document=**new** Document();

//6,创建field

//添加商品 id,title,sellPoint

Field idField=**new** LongField("id", 1, Store.***YES***);

Field titleField=**new**

TextField("title", "rose", Store.***YES***);

Field sellPointField=**new** TextField

("sellPoint", "i love you", Store.***YES***);

document.add(idField);

document.add(titleField);

document.add(sellPointField);

//7,写数据,进行中文分词和倒排索引

indexWriter.addDocument(document);

//8,关闭资源

indexWriter.close();

}

}

运行代码后，在项目根目录下生成index文件夹。

### 查看索引

双击“\培优课前资料\亿发,移动端课前资料\solr\查看索引工具\4.10.2\luke.bat”启动工具。

查看关键词, 如图- 103所示

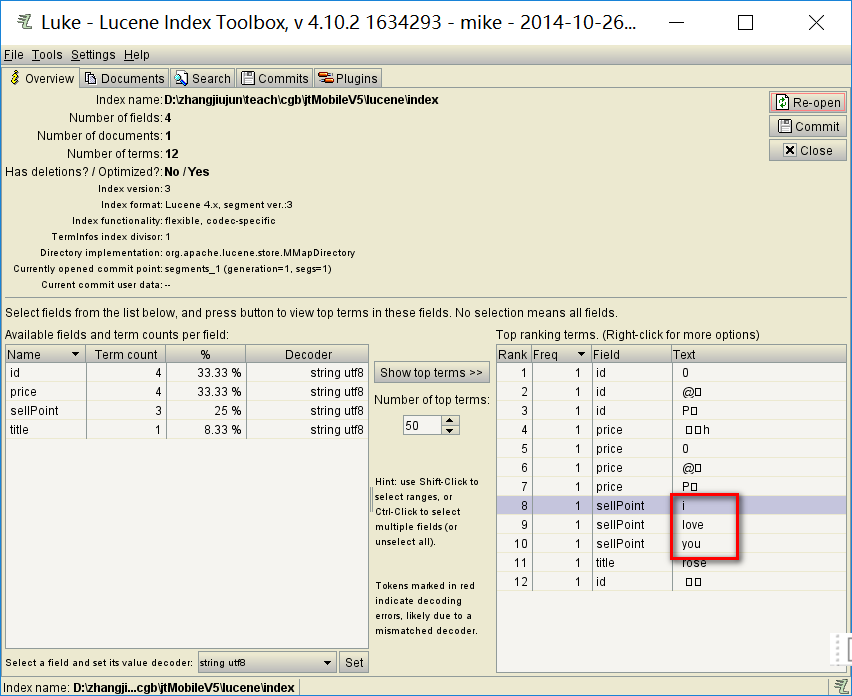


图- 103

查看记录数, 如图- 104所示

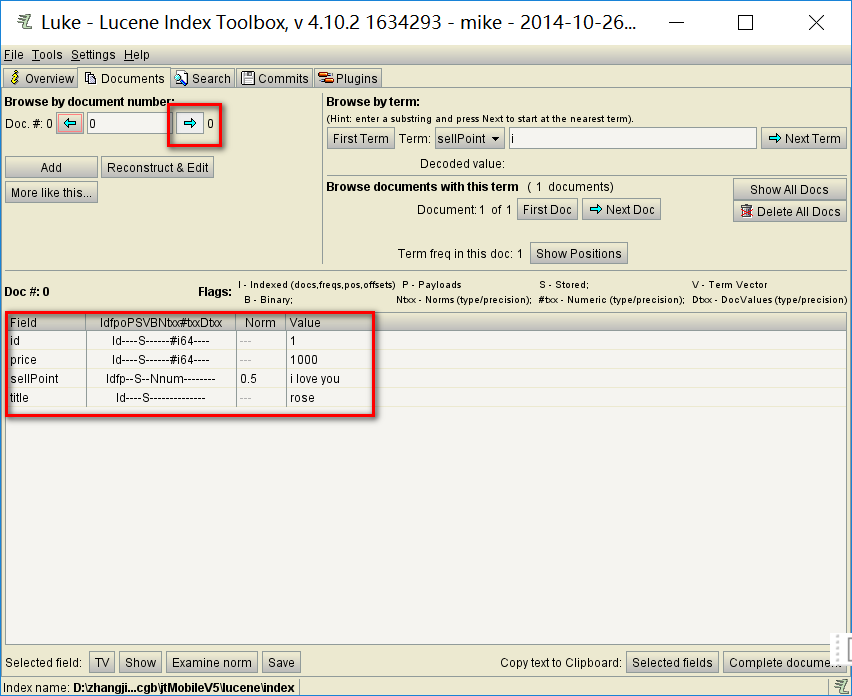


图- 104

直接查询, 如图- 105所示

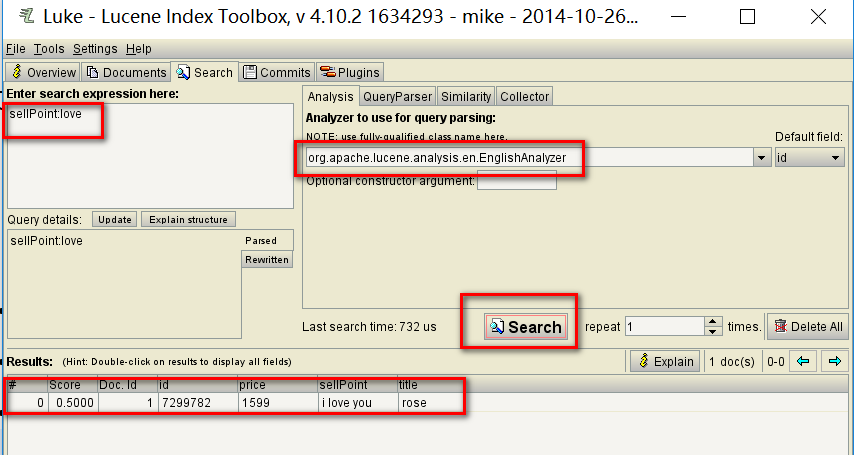


图- 105

再查询sellPoint:love

添加中文数据，能看关键词，记录，查询不了。

Textfield才会分词，stringfield 不会分词

### 根据索引实现检索

创建新的类，先查询love,再查询小米

**public** **class** SearchTest {

@Test

**public** **void** seracher() **throws** IOException{

//1,指定保存数据的文件夹

Directory directory = FSDirectory.*open*(**new** File("./index"));

//2,创建读对象

IndexReader indexReader=IndexReader.~~open~~(directory);

//3,创建检索对象

IndexSearcher indexSearcher = **new** IndexSearcher(indexReader);

//4,指定查询关键字

Query query = **new** TermQuery(**new** Term("sellPoint", "love"));

//Query query = new TermQuery(new Term("sellPoint", "小米"));

//5,执行查询

//查询 results 表示最顶端的20条记录数

TopDocs docs = indexSearcher.search(query, 20);

//访问总数

System.***out***.println("数据访问的总数:"+docs.totalHits);

//6遍历查询结果

//scoreDocs

//scoreDoc是个docuement

//scoreDoc是对document封装 例子 list,arrayList,linkedList

//score成绩

**for** ( ScoreDoc scoreDoc : docs.scoreDocs) {

//score ,doc.id,title,sellpoint

System.***out***.println("获取文章的得分:"+scoreDoc.score); //分数越高 越靠前

**int** index = scoreDoc.doc; //获取索引值

//获取文章内容

Document document = indexSearcher.doc(index);

//输出文章内容

System.***out***.println("标题:"+document.get("title"));

System.***out***.println("卖点:"+document.get("sellPoint"));

System.***out***.println("价格:"+document.get("price"));

System.***out***.println("id为:"+document.get("id"));

}

//7,关闭资源

directory.close();

}

}

### Lucene中存在问题

说明:

可以为商品记录生成索引文件.但是需要每条信息都进行遍历.并且将item对象转化为Document对象之后才能创建索引.

缺点:

1. 效率太低
2. 如果商品修改了,则需要同步索引文件.
3. 如果商品新增了,则需要重新遍历重新索引.
4. 重新生成索引文件的时间较长.

## Solr介绍

### 说明

Solr是一个独立的企业级搜索应用服务器，它对外提供类似于Web-service的API接口。用户可以通过http请求，向搜索引擎服务器提交一定格式的XML文件，生成索引；也可以通过Http Get操作提出查找请求，并得到XML格式的返回结果.

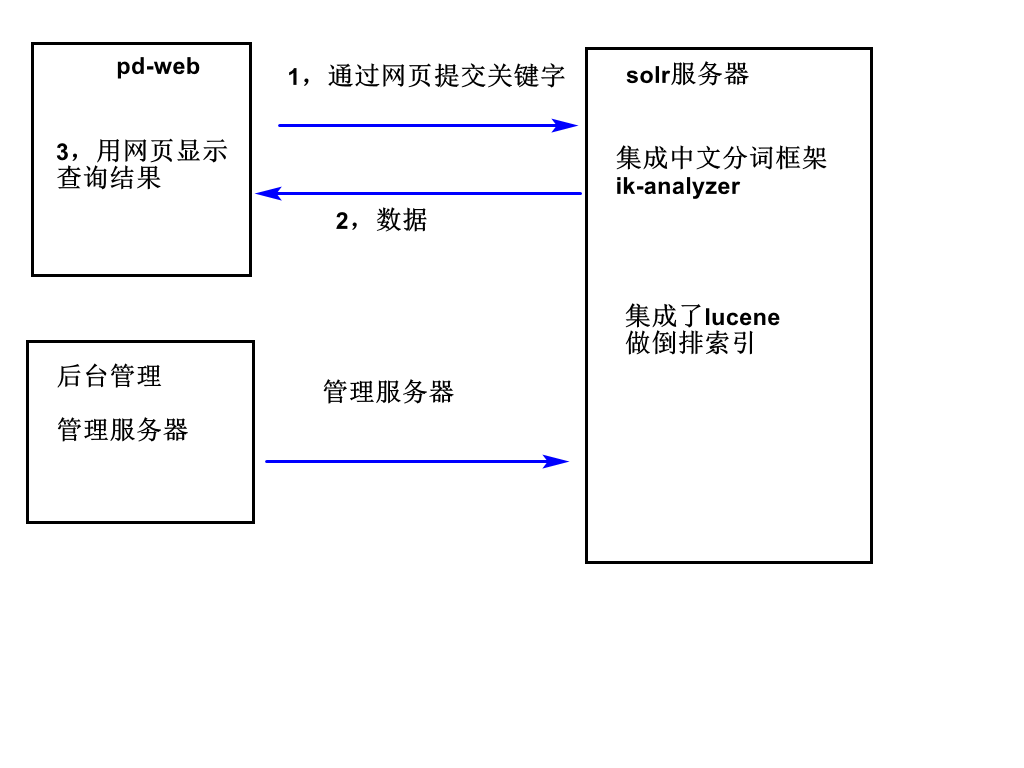
基于Lucene的全文搜索服务器。同时对其进行了扩展，提供了比Lucene更为丰富的查询语言，同时实现了可配置、可扩展并对查询性能进行了优化，并且提供了一个完善的功能管理界面，是一款非常优秀的全文搜索引擎。

特点:

1. solr可以根据数据库表自动生成索引文件.

2. Solr可以动态的定期自动更新索引(对更新的数据进行索引的修改)

### Solr与lucene,ikAnalyzer的关系



### 同类型产品

ElasticSearch是一个基于Lucene的搜索服务器。它提供了一个分布式多用户能力的全文搜索引擎。

网址:https://www.elastic.co/cn/products/elasticsearch, 如图- 106所示



图- 106

## Solr安装

### 环境准备

虚拟机使用京淘阶段的centos6.5,不用装新的

如果以前的虚拟机有问题，要装新的，请参考附录中的虚拟机安装

service iptables stop

### 上传安装包

Solr5.0以上版本要求jdk必须是1.8。

安装包文件是“\亿发,移动端课前资料\solr\solrLinux安装包”文件夹中的solr-5.2.1.tgz，上传到/usr/local/src文件夹中。然后解压文件。

tar -xzvf solr-5.2.1.tgz

### 启动solr

cd /usr/local/src/solr-5.2.1/bin

sh solr start

启动成功后如图-107 所示

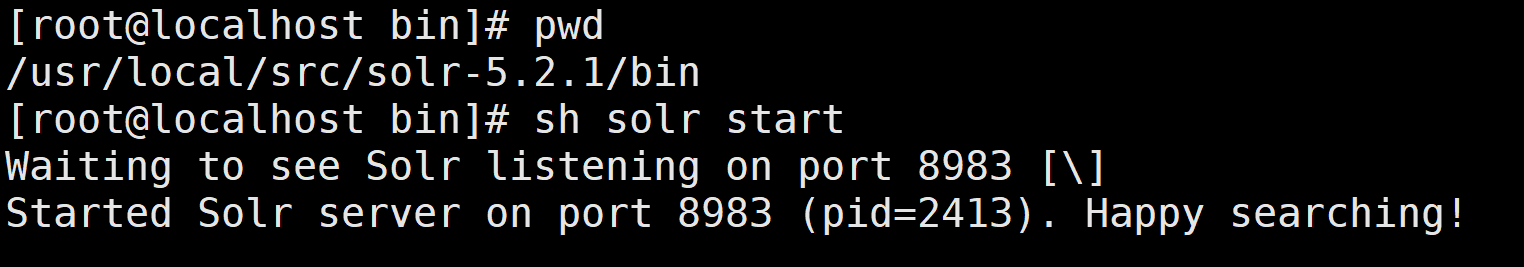


图- 107

启动后如果报错，用浏览器访问<http://ip:8983>，如果显示网页，说明成功了，或再执行sh solr start一次。或杀进程重启

### 访问后台管理

http://虚拟机ip:8983, 如图-108 所示

用chrome浏览器访问后台，用360浏览器访问有可能不显示界面。

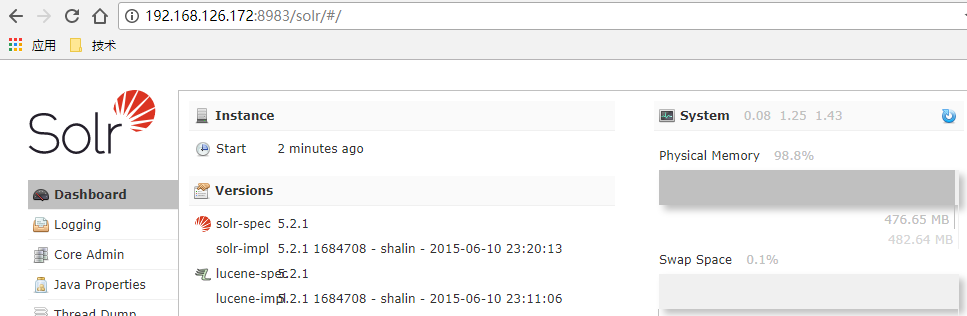


图- 108

## 添加core

### 需求

Mysql不支持中文分词，倒排索引，为了提高查询速度需要把mysql中的数据复制到solr服务器中, 如图-109 所示。

做查询为什么不是直接查询mysql?

因为mysql不支持中文分词，倒排索引

把mysql拷贝到solr中

在 拷贝的时候，solr用ik-analyzer进行中文分词

用lucene进行倒排索引

查询速度比mysql快

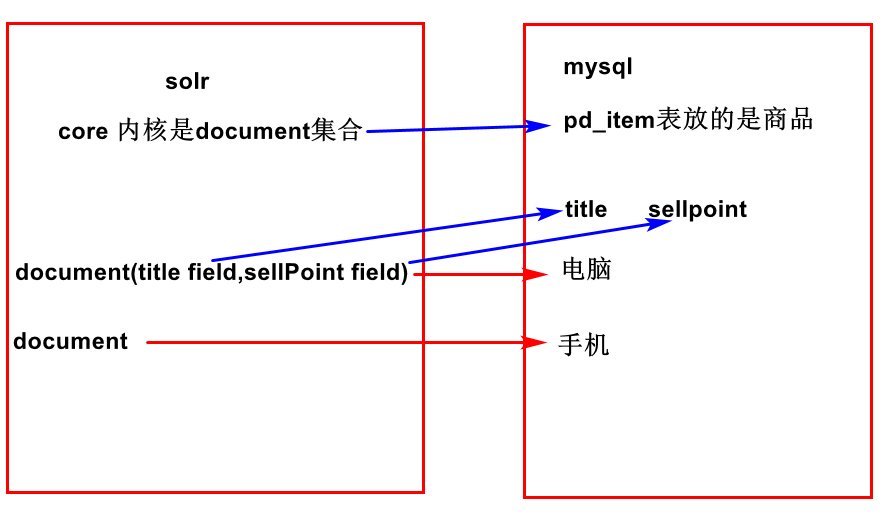


图- 109

### 什么是core

Core是多个document数据的集合。

### 创建core出错

单击在边的Core Admin添加core, 如图- 110所示

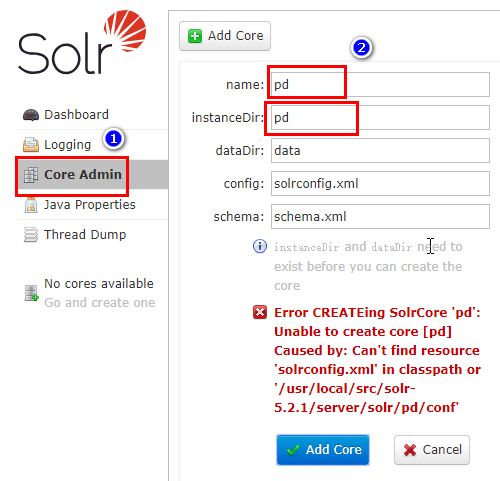


图- 110

添加pd 会报下面错误, 如图-111 所示

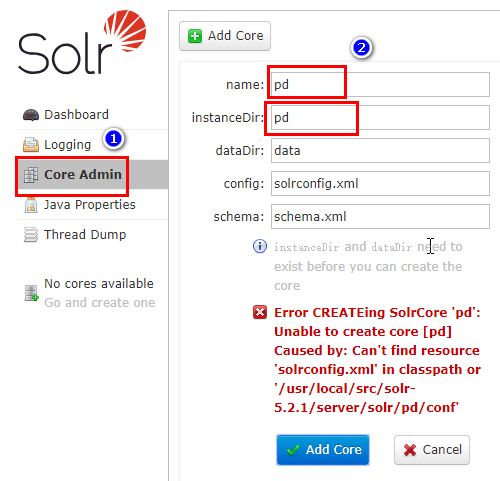


图- 111

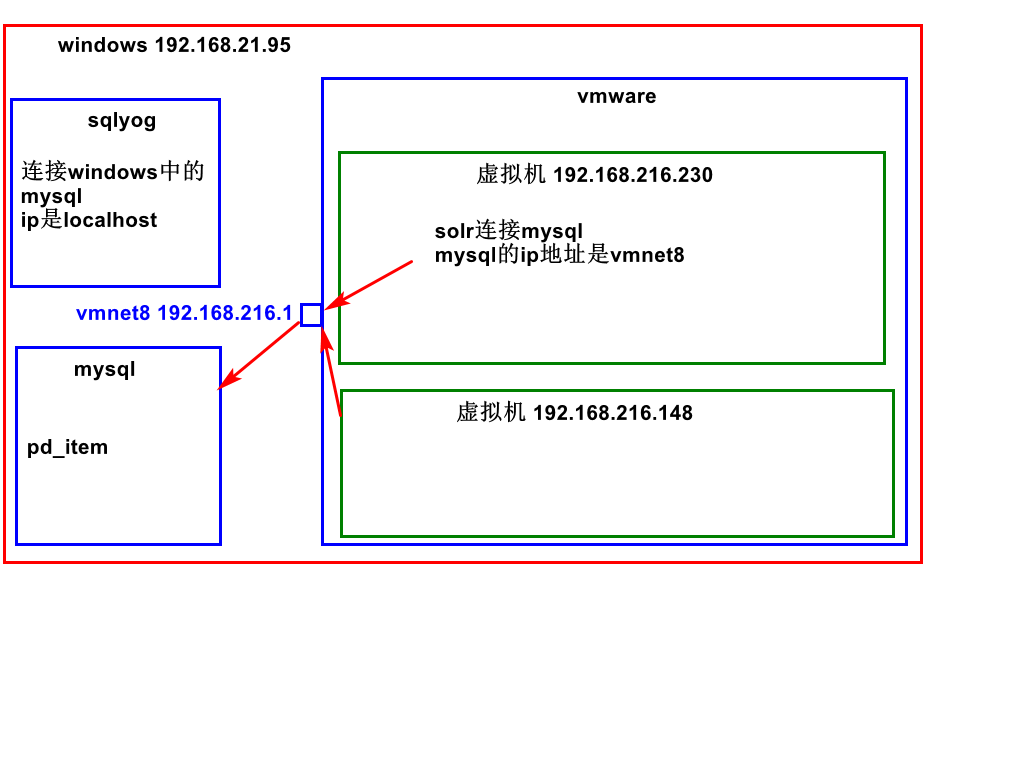
### 创建pd,pd/conf,pd/data

1. 在/usr/local/src/solr-5.2.1/server/solr目录下新建pd pd/conf pd/data三个文件夹

cd /usr/local/src/solr-5.2.1/server/solr

mkdir pd pd/conf pd/data

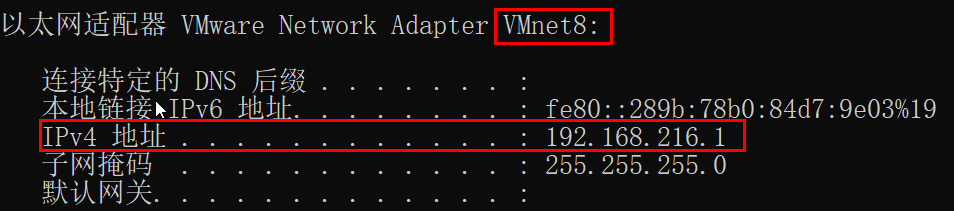
### 修改配置文件并上传



1. 修改“\亿发,移动端课前资料\solr\solrLinux安装包\02-所需配置文件\pd\conf\dih-config.xml”

把mysql的ip地址改成windows中vmnet8的地址,

Windowns中进入dos执行ipconfig,查看vmnet8的地址



<dataConfig>

<dataSource type="JdbcDataSource"

driver="com.mysql.jdbc.Driver"

url="jdbc:mysql://192.168.216.1:3306/pd\_store"

user="root"

password="root"

batchSize="100"

autoCommit="false" />

</dataConfig>

第一条select语句执行一次把mysql中的数据拷贝到solr中

第二条select语句每隔8小时执行一次，把mysql中最新数据，修改过的数据拷贝到solr中,

<dataConfig>

<document name="item">

<entity name="item" pk="id"

query="SELECT id,title,sell\_point,price,barcode,image,cid,`status`,created,updated FROM pd\_item"

deltaQuery="SELECT id,title,sell\_point,price,barcode,image,cid,`status`,created,updated FROM pd\_item WHERE updated &gt; date\_add(str\_to\_date('${dih.last\_index\_time}','%Y-%m-%d %H:%i:%s'),interval 8 hour)"

transformer="RegexTransformer">

</entity>

</document>

</dataConfig>

2.查看schema.xml

作用:

* 定义ik分词器
* 定义field

不用修改。

<?xml version="1.0" encoding="UTF-8" ?>

<schema name="example" version="1.5">

<field name="\_version\_" type="long" indexed="true" stored="true"/>

<field name="\_root\_" type="string" indexed="true" stored="false"/>

<field name="id" type="long" indexed="true" stored="true" required="true" multiValued="false" />

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

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

<field name="price" type="long" indexed="true" stored="true"/>

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

<field name="created" type="date" indexed="true" stored="true"/>

<field name="updated" type="date" indexed="true" stored="true"/>

<uniqueKey>id</uniqueKey>

<fieldType name="string" class="solr.StrField" sortMissingLast="true" />

<fieldType name="boolean" class="solr.BoolField" sortMissingLast="true"/>

<fieldType name="int" class="solr.TrieIntField" precisionStep="0" positionIncrementGap="0"/>

<fieldType name="float" class="solr.TrieFloatField" precisionStep="0" positionIncrementGap="0"/>

<fieldType name="long" class="solr.TrieLongField" precisionStep="0" positionIncrementGap="0"/>

<fieldType name="double" class="solr.TrieDoubleField" precisionStep="0" positionIncrementGap="0"/>

<fieldType name="date" class="solr.TrieDateField" precisionStep="0" positionIncrementGap="0"/>

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

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

</fieldType>

</schema>

3.查看solrConfig.xml

定义了依赖的lucene的版本。不用改。

<luceneMatchVersion>4.10.2</luceneMatchVersion>

4.把conf文件夹中文件上传到/usr/local/src/solr-5.2.1/server/solr/pd/conf

### 为IK分词器添加索引词



\亿发,移动端课前资料\solr\solrLinux安装包\02-所需配置文件\classes\ik\_ext.dic文件存放新添加的索引词

ik\_stopwords.dic 存放停止的索引词

把“\亿发,移动端课前资料\solr\solrLinux安装包\02-所需配置文件”中的classes文件夹拷贝到“/usr/local/src/solr-5.2.1/server/solr-webapp/webapp/WEB-INF”中, 如图- 114所示

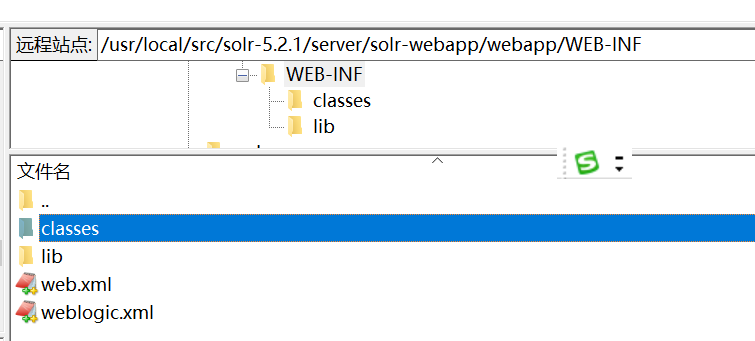


图- 114

### 添加jar包文件

把“亿发,移动端课前资料\solr\solrLinux安装包\02-所需配置文件\solr配置jar包”中的4个jar文件拷贝到“/usr/local/src/solr-5.2.1/server/solr-webapp/webapp/WEB-INF/lib”文件夹, 如图- 115所示

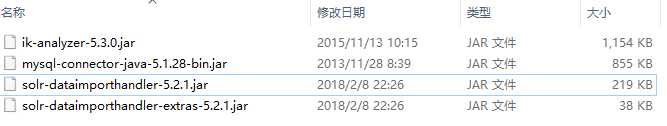


图- 115

### 重启solr

cd /usr/local/src/solr-5.2.1/bin

sh solr restart

命令执行成功后如图-116 所示

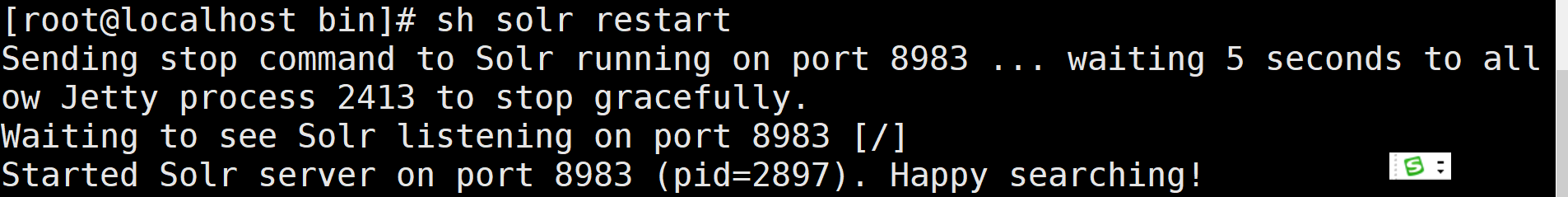


图- 116

### 重新添加core

访问http://虚拟机ip:8983,重新添加pd, 如图- 117所示

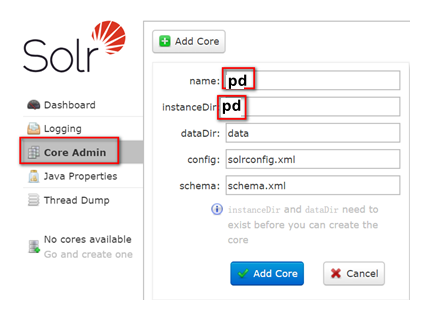


图- 117

### Mysql上授权

grant [权限] on [数据库名].[表名] to ['用户名']@['web服务器的ip地址'] identified by ['密码'];

进入sqlyog执行下面的命令

grant all on \*.\* to 'root'@'%' identified by 'root';

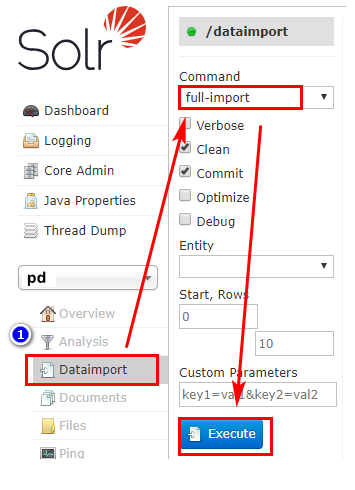
### 关闭windows防火墙

打开“网络和internet设置”—》找到以态网网卡🡪windwos防火墙🡪把域网络，专用网络，公用网络的防火墙关闭。

### 执行导入数据

http://虚拟机ip:8983

选中pd core—> dataimport 🡪full import🡪execute



过30秒再点Refresh status，会显示添加了多少个数据, 如图-118 所示

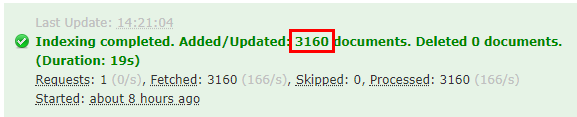


图- 118

导入数据完后，过30秒再点overview，会显示添加的数据行数如图-119 所示.

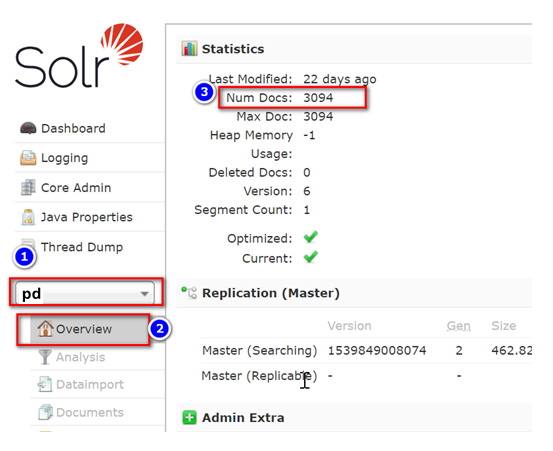
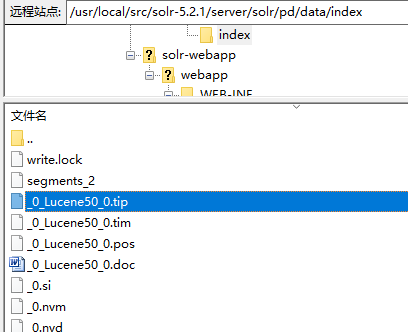


图- 119

生成的索引文件在/usr/local/src/solr-5.2.1/server/solr/pd/data/index中



### 进行查询, 如图-120 所示



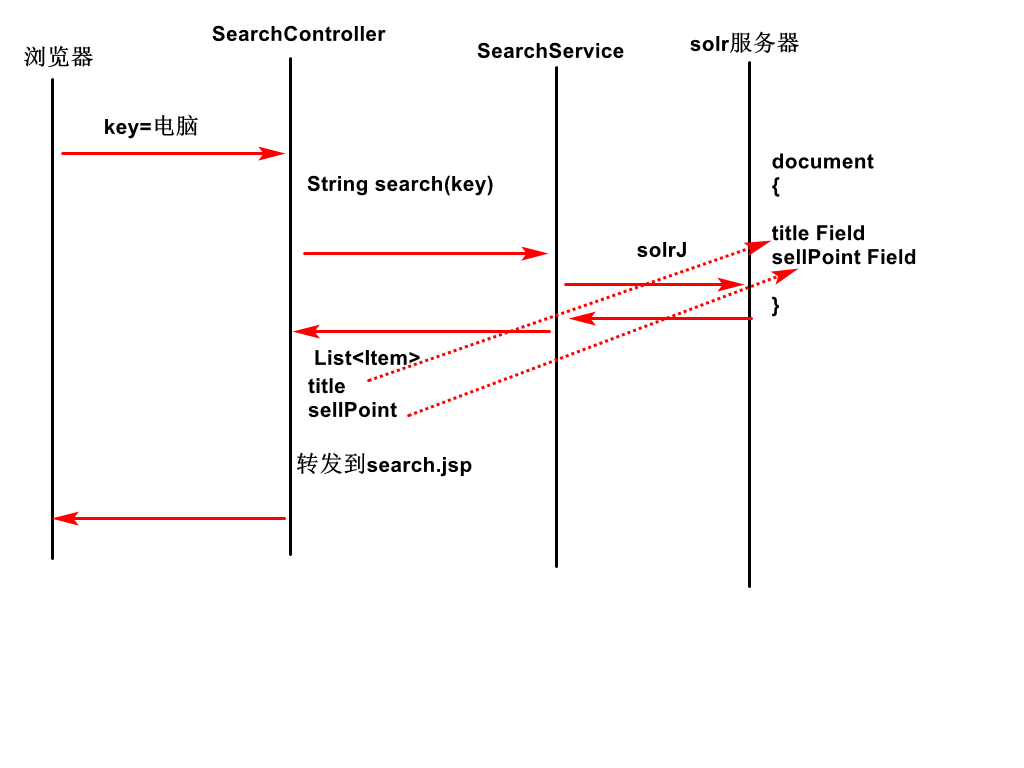
图- 120

# 拼多实现全文检索

## 需求

查询商品信息

## 分析



需要实体类Item,接口SearchService,接口实现类SearchServiceImpl

## 整合solrJ

### 添加依赖

<!-- solrj -->

<dependency>

<groupId>org.springframework.data</groupId>

<artifactId>spring-data-solr</artifactId>

<version>1.5.4.RELEASE</version>

</dependency>

<dependency>

<groupId>org.apache.solr</groupId>

<artifactId>solr-solrj</artifactId>

<version>5.2.1</version>

</dependency>

### 添加solr服务器信息

application.yml中增加黄色背景内容，修改ip地址



spring:

datasource:

type: com.alibaba.druid.pool.DruidDataSource

driver-class-name: com.mysql.jdbc.Driver

url: jdbc:mysql://127.0.0.1:3306/pd\_store

username: root

password: root

data:

solr:

host: http://192.168.216.230:8983/solr/pd

### 创建实体类

在com.pd.pojo中创建Item实体类

//实体类，对应solr中的数据

//<field title,id,sellPoing,image

**public** **class** Item **implements** Serializable

{

//来之solr中的id field

@Field("id")

**private** Long id;

@Field("title")

**private** String title;

@Field("sellPoint")

**private** String sellPoint;

@Field("price")

**private** Long price;

@Field("image")

**private** String image;

**public** String getImage() {

**return** image;

}

**public** **void** setImage(String image) {

**this**.image = image;

}

**public** Long getId() {

**return** id;

}

**public** **void** setId(Long id) {

**this**.id = id;

}

**public** String getTitle() {

**return** title;

}

**public** **void** setTitle(String title) {

**this**.title = title;

}

**public** String getSellPoint() {

**return** sellPoint;

}

**public** **void** setSellPoint(String sellPoint) {

**this**.sellPoint = sellPoint;

}

**public** Long getPrice() {

**return** price;

}

**public** **void** setPrice(Long price) {

**this**.price = price;

}

}

### 创建接口

在com.pd.service包中创建接口

**public** **interface** SearchService {

**public** List<Item> findItemByKey(String key);

}

@Service

**public** **class** SearchServiceImpl **implements** SearchService {

@Autowired

SolrClient solrClient;

**public** List<Item> findItemByKey(String key) {

**try** {

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

query.setStart(0);// 从那一行开始取数据

query.setRows(20);// 取20行。

QueryResponse response = solrClient.query(query);

List<Item> itemList = response.getBeans(Item.**class**);

**return** itemList;

} **catch** (Exception e) {

e.printStackTrace();

}

**return** **null**;

}

}

### 编写Searchcontroller

@Controller

**public** **class** SearchController {

@Autowired

SearchService searchService;

@RequestMapping("/search/toSearch.html")

**public** String search(String key,Model model)

**throws** Exception

{

List<Item> itemList=

searchService.findItemByKey(key);

model.addAttribute("list", itemList);

//转发到search.jsp

**return** "/search.jsp";

}

}

### 效果展现

1.直接在浏览器地址栏中输入下面的url

<http://www.pd.com/search/toSearch.html?key=电脑>

2.通过网页访问,在输入框中输入关键词进行搜索, 如图-125 所示



图- 125

## 使用mysql和solr进行查询对比

使用solr比mysql快

Solr的数据来之mysql

# 总结

中文分词是ikAanlyzer实现的

文档表

|  |  |
| --- | --- |
| id | 网页的url |
| 1 | 1.html(java是编程语言) |
| 2 | 2.html(java开发工程师) |

当第一次添加数据，抓到网页就进行中文分词

索引表

|  |  |
| --- | --- |
| 关键字 | 所有文档 |
| java | 1,2 |
| 开发工程师 | 2 |

## 分布式锁

setnx 1022 lock

(integer) 1

setnx 1022 lock

(integer) 0

del 1022 lock

(integer) 1

setnx 1022 lock

(integer) 1

expire 1022 30

(integer) 1

ttl 1022

(integer) 24