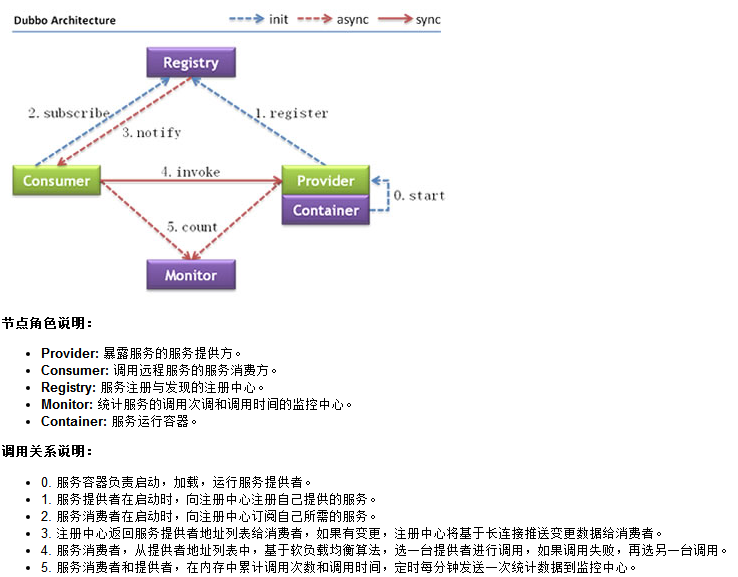
[dubbo+zookeeper例子](http://www.cnblogs.com/huqianliang/p/5664440.html)

0.原理

Alibaba有好几个分布式框架，主要有：进行远程调用(类似于RMI的这种远程调用)的(dubbo、hsf)，jms消息服务(napoli、notify)，KV数据库(tair)等。这个框架/工具/产品在实现的时候，都考虑到了容灾，扩展，负载均衡，于是出现一个配置中心(ConfigServer)的东西来解决这些问题。

基本原理如图：



在我们的系统中，经常会有一些跨系统的调用，如在A系统中要调用B系统的一个服务，我们可能会使用RMI直接来进行，B系统发布一个RMI接口服务，然后A系统就来通过RMI调用这个接口，为了解决容灾，扩展，负载均衡的问题，我们可能会想很多办法，alibaba的这个办法感觉不错。

本文只说dubbo，原理如下：

* **ConfigServer**

配置中心，和每个Server/Client之间会作一个实时的心跳检测（因为它们都是建立的Socket长连接），比如几秒钟检测一次。收集每个Server提供的服务的信息，每个Client的信息，整理出一个服务列表，如：

|  |  |  |
| --- | --- | --- |
| **serviceName** | **serverAddressList** | **clientAddressList** |
| UserService | 192.168.0.1，192.168.0.2，192.168.0.3，192.168.0.4 | 172.16.0.1，172.16.0.2 |
| ProductService | 192.168.0.3，192.168.0.4，192.168.0.5，192.168.0.6 | 172.16.0.2，172.16.0.3 |
| OrderService | 192.168.0.10，192.168.0.12，192.168.0.5，192.168.0.6 | 172.16.0.3，172.16.0.4 |

当某个Server不可用，那么就更新受影响的服务对应的**serverAddressList**，即把这个Server从**serverAddressList**中踢出去（从地址列表中删除），同时将推送**serverAddressList**给这些受影响的服务的clientAddressList里面的所有Client。如：192.168.0.3挂了，那么UserService和ProductService的**serverAddressList**都要把192.168.0.3删除掉，同时把新的列表告诉对应的Client 172.16.0.1，172.16.0.2，172.16.0.3；

当某个Client挂了，那么更新受影响的服务对应的**clientAddressList**

ConfigServer根据服务列表，就能提供一个web管理界面，来查看管理服务的提供者和使用者。

新加一个Server时，由于它会主动与ConfigServer取得联系，而ConfigServer又会将这个信息主动发送给Client，所以新加一个Server时，只需要启动Server，然后几秒钟内，Client就会使用上它提供的服务

* **Client**

调用服务的机器，每个Client启动时，主动与ConfigServer建立Socket长连接，并将自己的IP等相应信息发送给ConfigServer。

Client在使用服务的时候根据服务名称去ConfigServer中获取服务提供者信息（这样ConfigServer就知道某个服务是当前哪几个Client在使用），Client拿到这些服务提供者信息后，与它们都建立连接，后面就可以直接调用服务了，当有多个服务提供者的时候，Client根据一定的规则来进行负载均衡，如轮询，随机，按权重等。

一旦Client使用的服务它对应的服务提供者有变化（服务提供者有新增，删除的情况），ConfigServer就会把最新的服务提供者列表推送给Client，Client就会依据最新的服务提供者列表重新建立连接，新增的提供者建立连接，删除的提供者丢弃连接

* **Server**

真正提供服务的机器，每个Server启动时，主动与ConfigServer建立Scoket长连接，并将自己的IP，提供的服务名称，端口等信息直接发送给ConfigServer，ConfigServer就会收集到每个Server提供的服务的信息。

优点：

1，只要在Client和Server启动的时候，ConfigServer是好的，服务就可调用了，如果后面ConfigServer挂了，那只影响ConfigServer挂了以后服务提供者有变化，而Client还无法感知这一变化。

2，Client每次调用服务是不经过ConfigServer的，Client只是与它建立联系，从它那里获取提供服务者列表而已

3，调用服务-负载均衡：Client调用服务时，可以根据规则在多个服务提供者之间轮流调用服务。

4，服务提供者-容灾：某一个Server挂了，Client依然是可以正确的调用服务的，当前提是这个服务有至少2个服务提供者，Client能很快的感知到服务提供者的变化，并作出相应反应。

5，服务提供者-扩展：添加一个服务提供者很容易，而且Client会很快的感知到它的存在并使用它。

1.开发软件、资料

* jdk1.7.0\_79 ,安装并配置好java开发环境
* zookeeper-3.4.5   下载地址：http://download.csdn.net/detail/adam\_zs/9470314
* Tomcat 7.0 配置入eclipse或者myeclipse都可以
* dubbo-admin-2.5.3.war  下载地址：http://download.csdn.net/detail/adam\_zs/9470323
* apache-maven-3.2.5  配置入eclipse或者myeclipse都可以
* dubbo官方文档 http://dubbo.io/Home-zh.htm
* Dubbo安装 下载地址：<https://github.com/alibaba/dubbo/releases>  pom.xml:<http://files.cnblogs.com/files/belen/pom.xml>

2.关键步骤

zookeeper安装部署（

ZooKeeper是一个分布式的，开放源码的分布式应用程序协调服务，是Google的Chubby一个开源的实现，是Hadoop和Hbase的重要组件。实例中，zookeeper将作为dubbo服务的注册中心。同时负责集群调度。

为什么要用zookeeper?

Zookeeper可以提供配置管理、命名服务、分布式算法、集群管理功能。具体说明参看如下文章：

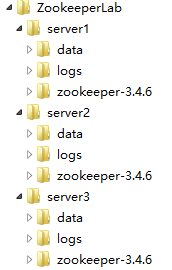
<http://zhidao.baidu.com/question/919262980452730419.html?fr=iks&word=zookeeper+dubbo+%B9%D8%CF%B5&ie=gbk>

）

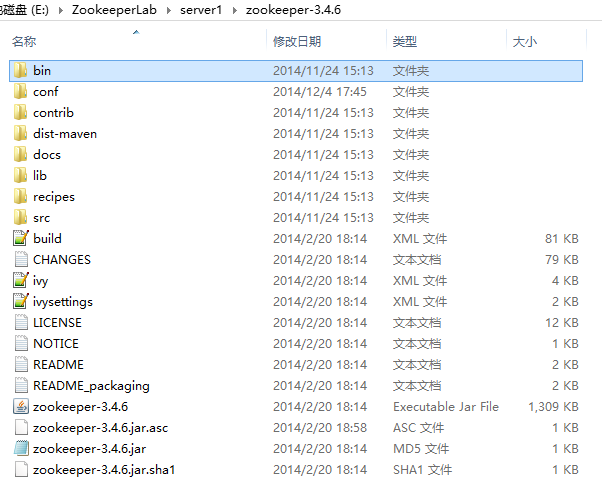
Zookeeper部署

1、dubbo依赖于**Zookeeper**，实现任务的分布式配置及各服务间的交互通信，Zookeeper以TreeNode类型进行存储，支持Cluster形式部署且保证最终数据一致性，关于ZK的资料网上比较丰富，相关概念不再重复介绍，本文以zookeeper-3.4.6为例，请从官网下载[http://zookeeper.apache.org](http://zookeeper.apache.org/)。

2、创建ZookeeperLab文件夹目录，模拟部署3台Zookeeper服务器集群，目录结构如下。



3、解压从官网下载的zookeeper-3.4.6.tar文件，并分别复制到三台ZkServer的zookeeper-3.4.6文件夹。

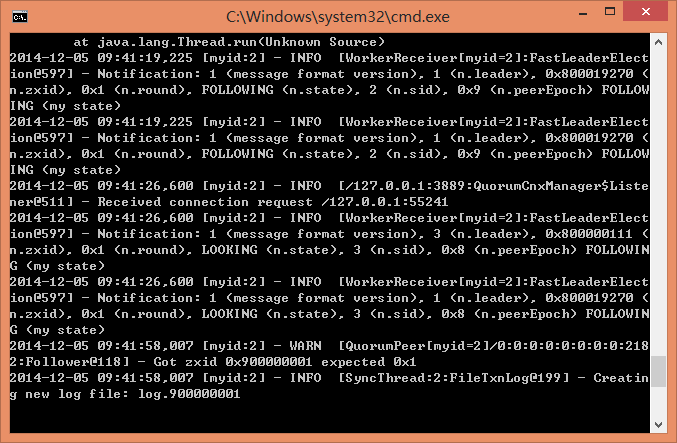


4、分别在三台ZkServer的data目录下创建myid文件（注意没有后缀），用于标识每台Server的ID，在Server1\data\myid文件中保存单个数字1，Server2的myid文件保存2，Server3的myid保存3。

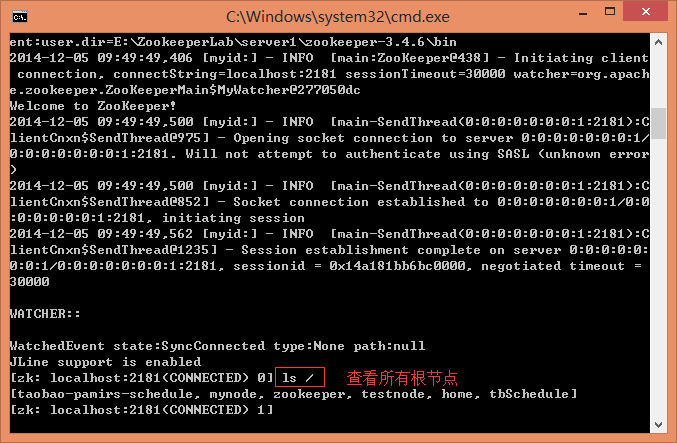
5、创建ZkServer的配置文件，在zookeeper-3.4.6\conf文件夹目录下创建zoo.cfg，可以从示例的zoo\_sample.cfg 复制重命名。因为在同一台机器模拟Cluster部署，端口号不能重复，配置文件中已经有详细的解释，修改后的配置如下，其中Server1端口号2181，Server2端口号2182，Server3端口号2183。

http://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

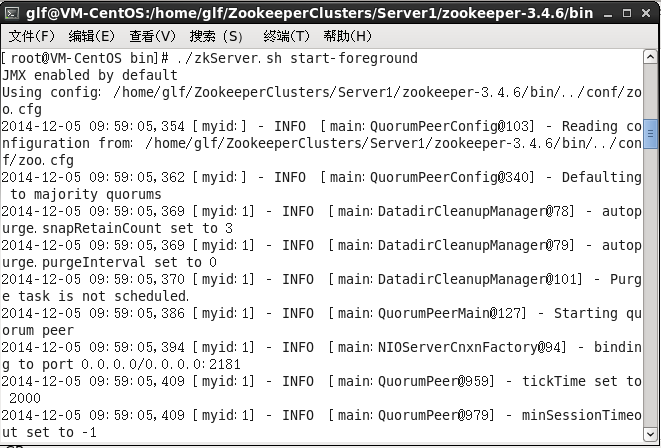
6、通过zookeeper-3.4.6\bin文件夹zkServer.bat文件启动ZKServer，由于Cluster部署需要选举Leader和Followers，所以在3个ZKServer全部启动之前会提示一个WARN，属正常现象。



7、Zookeeper启动成功后可以通过zookeeper-3.4.6\bin文件夹的 zkCli.bat验证连接是否正常，比如创建节点“create /testnode helloworld”，查看节点“get /testnode”，连接到组群中其它ZkServer，节点数据应该是一致的。更多指令请使用help命令查看。



8、对于Linux环境下部署基本一致，zoo.cfg配置文件中data和datalog文件夹路径改为linux格式路径，使用“**./zkServer.sh** **start-foreground**”命令启动ZkServer，注意start启动参数不能输出异常信息。



9、至此Zookeeper的配置完毕。

dubbo治理平台部署（

上面内容看起来没那么直观。如果有一个控制台来管理和展现就太棒了。不得不说dubbo还是挺贴心的。

下载

官网下载地址：

<http://code.alibabatech.com/mvn/releases/com/alibaba/dubbo-admin/2.4.1/dubbo-admin-2.4.1.war>

但是该地址最近一直无法下载。

http://pan.baidu.com/share/link?shareid=2205137761&uk=3442370350&fid=707816148751698 可以通过这里下载。

安装

将war包拷贝到tomcat/webapps目录下，启动tomcat。浏览器中输入：

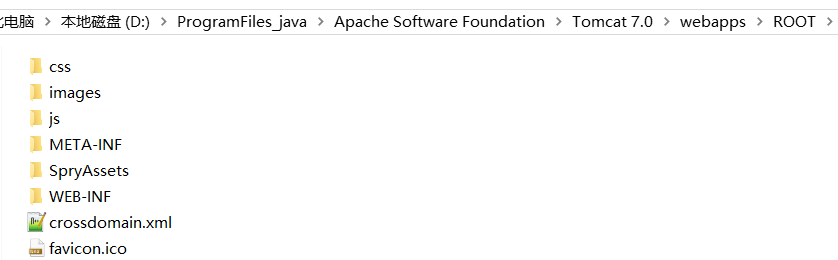
<http://localhost:8080/dubbo/>

）

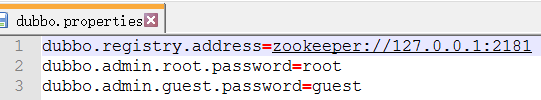
dubbo-admin-2.5.3.war解压后得到如下文件



删除D:\ProgramFiles\_java\Apache Software Foundation\Tomcat 7.0\webapps\ROOT路径下所有文件，复制解压文件到该路径，效果如下



如需要修改登陆dubbo治理平台密码，进入D:\ProgramFiles\_java\Apache Software Foundation\Tomcat 7.0\webapps\ROOT\WEB-INF路径，打开dubbo.properties



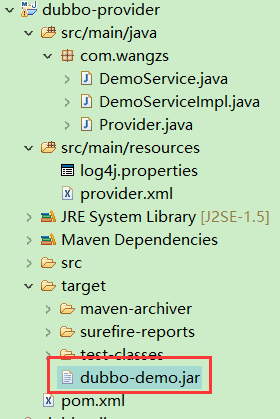
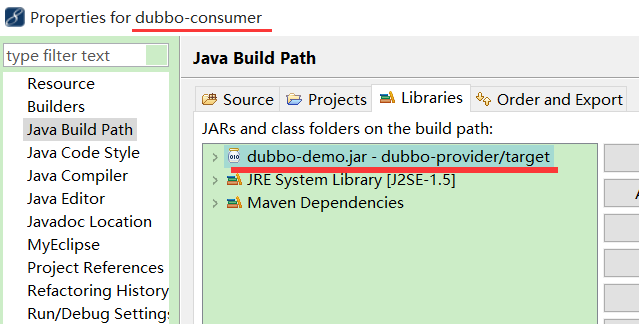
默认两个用户，用户名密码分别为 root/root  guest/guest

启动tomcat，浏览器输入地址：http://localhost:8080/ 进入dubbo治理平台



出现上图说明dubbo治理平台部署完毕

服务提供者打包给服务消费者引用

3.源码

<!-- zookeeper -->

<dependency>

<groupId>com.github.sgroschupf</groupId>

<artifactId>zkclient</artifactId>

<version>0.1</version>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>org.apache.zookeeper</groupId>

<artifactId>zookeeper</artifactId>

<version>3.4.6</version>

<exclusions>

<exclusion>

<artifactId>jmxtools</artifactId>

<groupId>com.sun.jdmk</groupId>

</exclusion>

<exclusion>

<artifactId>jmxri</artifactId>

<groupId>com.sun.jmx</groupId>

</exclusion>

<exclusion>

<artifactId>jms</artifactId>

<groupId>javax.jms</groupId>

</exclusion>

</exclusions>

</dependency>

[复制代码](javascript:void(0);)

复制代码

<!-- dubbo-->

<dependency>

<groupId>com.alibaba</groupId>

<artifactId>dubbo</artifactId>

<version>2.4.9</version>

<scope>compile</scope>

<exclusions>

<exclusion>

<artifactId>spring</artifactId>

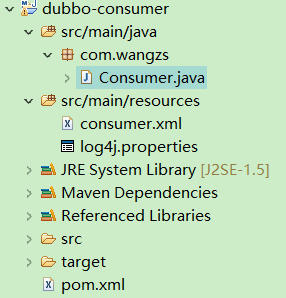
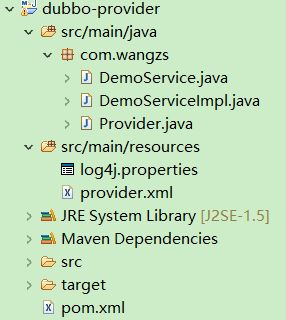
<groupId>org.springframework</groupId>

</exclusion>

</exclusions>

</dependency>

[复制代码](javascript:void(0);)

Provider applicationContext.xml:

<beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:dubbo="http://code.alibabatech.com/schema/dubbo"xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd http://code.alibabatech.com/schema/dubbo http://code.alibabatech.com/schema/dubbo/dubbo.xsd ">

<!-- 具体的实现bean -->

<bean id="demoService" class="com.unj.dubbotest.provider.impl.DemoServiceImpl"/>

<!-- 提供方应用信息，用于计算依赖关系 -->

<dubbo:application name="xixi\_provider"/>

<!--

使用multicast广播注册中心暴露服务地址 <dubbo:registry address="multicast://224.5.6.7:1234" />

-->

<!-- 使用zookeeper注册中心暴露服务地址 -->

<dubbo:registry protocol="zookeeper" address="127.0.0.1:2181"/>

<!-- 用dubbo协议在20880端口暴露服务 -->

<dubbo:protocol name="dubbo" port="2090"/>

<!-- 声明需要暴露的服务接口 -->

<dubbo:service interface="com.unj.dubbotest.provider.DemoService" ref="demoService"/>

</beans>

服务消费方代码dubbo-consumer  下载地址：http://download.csdn.net/detail/adam\_zs/9470354

Consumer applicationContext.xml:

<beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:dubbo="http://code.alibabatech.com/schema/dubbo"xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd http://code.alibabatech.com/schema/dubbo http://code.alibabatech.com/schema/dubbo/dubbo.xsd ">

<!-- 消费方应用名，用于计算依赖关系，不是匹配条件，不要与提供方一样 -->

<dubbo:application name="hi\_consumer"/>

<!-- 使用zookeeper注册中心暴露服务地址 -->

<!--

<dubbo:registry address="multicast://224.5.6.7:1234" />

-->

<dubbo:registry protocol="zookeeper" address="127.0.0.1:2181"/>

<!-- 生成远程服务代理，可以像使用本地bean一样使用demoService -->

<dubbo:reference id="demoService" interface="com.unj.dubbotest.provider.DemoService"/>

</beans>

调用方式 注入spring后，通过ApplicationContext获取对应服务接口，调用服务方法。

4.启动顺序

1. 启动zookeeper
2. 启动tomcat，启动完毕可以输入地址http://localhost:8080/可以看到dubbo治理平台
3. 启动服务提供者 Provider.java
4. 启动消费者 Consumer.java

5.成功启动截图





