# 微服务架构设计

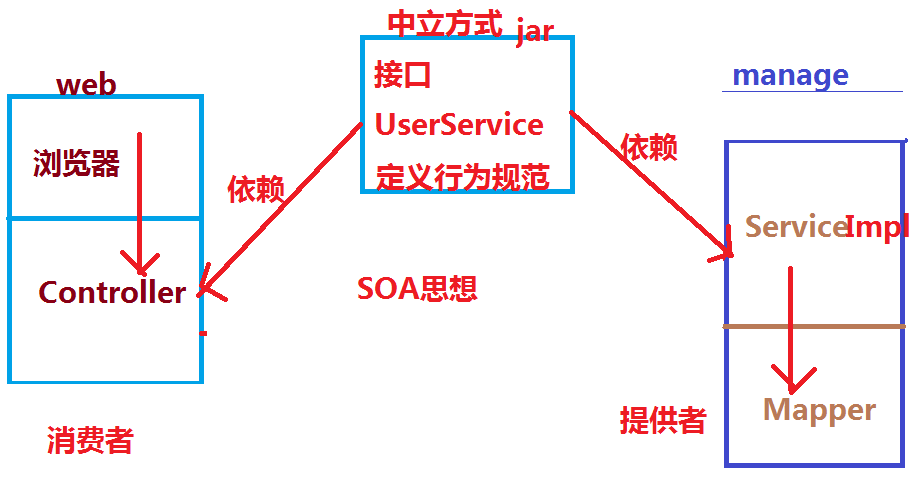
## 什么是微服务

### 概念

在**分布式**的基础之上.服务可以**独立的运行**.当服务发生故障时,可以实现故障的自动化迁移.用户无需关注.

### SOA思想

**面向服务的架构（SOA）**是一个组件模型，它将应用程序的不同功能单元（称为服务）进行拆分，并通过这些服务之间定义良好的**接口**和契约联系起来。接口是采用**中立**的方式进行定义的，它应该独立于实现服务的硬件平台、操作系统和编程语言。这使得构建在各种各样的系统中的服务可以以一种统一和通用的方式进行交互。



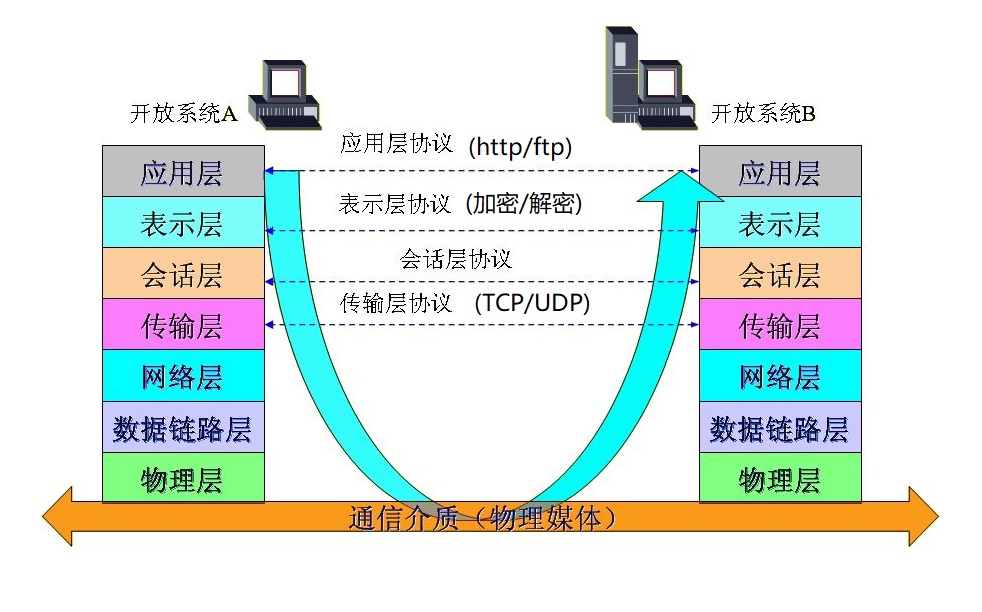
### RPC协议

RPC（Remote Procedure Call）—[远程过程调用](https://baike.baidu.com/item/%E8%BF%9C%E7%A8%8B%E8%BF%87%E7%A8%8B%E8%B0%83%E7%94%A8/7854346)，它是一种通过[网络](https://baike.baidu.com/item/%E7%BD%91%E7%BB%9C/143243)从远程计算机程序上请求服务，而不需要了解底层网络技术的协议。[RPC协议](https://baike.baidu.com/item/RPC%E5%8D%8F%E8%AE%AE)假定某些[传输协议](https://baike.baidu.com/item/%E4%BC%A0%E8%BE%93%E5%8D%8F%E8%AE%AE/8048821)的存在，如TCP或UDP，为通信程序之间携带信息数据。在OSI[网络通信](https://baike.baidu.com/item/%E7%BD%91%E7%BB%9C%E9%80%9A%E4%BF%A1/9636548)模型中，RPC跨越了[传输层](https://baike.baidu.com/item/%E4%BC%A0%E8%BE%93%E5%B1%82/4329536)和[应用层](https://baike.baidu.com/item/%E5%BA%94%E7%94%A8%E5%B1%82/4329788)。RPC使得开发包括网络[分布式](https://baike.baidu.com/item/%E5%88%86%E5%B8%83%E5%BC%8F)多程序在内的应用程序更加容易。

### OSI模型(网络7层协议规范)

网络7层协议如图所示.

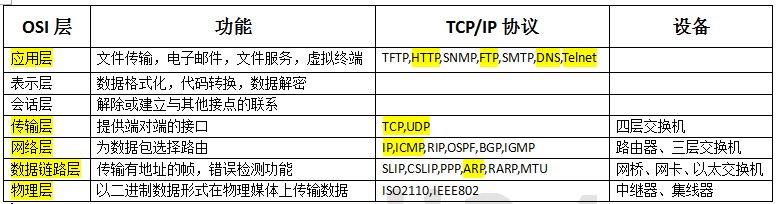
如图-22所示



**图-**22

层级关系与对应的协议.

如图-23所示



**图-**23

### RPC与HTTP区别

区别:

1.RPC是传输层协议(4层).而HTTP协议是应用层协议(7层).

2.RPC协议可以直接调用中立接口,HTTP协议不可以.

3.RPC通信协议是长链接,HTTP协议一般采用短连接需要3次握手(可以配置长链接添加请求头Keep-Alive: timeout=20).

(长连接，指在一个连接上可以连续发送多个[数据包](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%8C%85/489739" \t "_blank)，在连接保持期间，如果没有数据包发送，需要双方发链路检测包。)

4.RPC协议传递数据是加密压缩传输.HTTP协议需要传递大量的请求头信息.

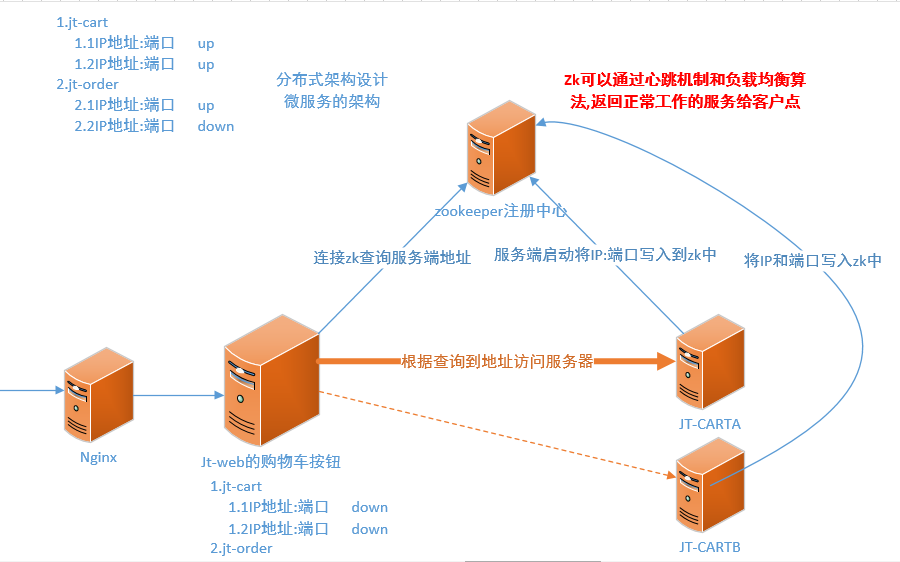
5.RPC协议一般都有**注册中心**.有丰富的监控机制.

### RPC总结

RPC叫远程过程调用.它是OSI模型中第四层协议.封装了四层以下的通讯方式.使用户不需要了解tcp/udp等网络传输协议,也能实现数据的传输.同时RPC一般使用时需要配置注册中心.

## 注册中心

### 注册中心工作原理



1. 当服务启动时,将服务信息服务名称/IP/端口写入注册中心.
2. 注册中心接收服务端信息时保存服务信息,并且维护服务列表数据
3. 当服务消费者启动时会通过IP:端口(注册中心)远程链接注册中心.

获取服务列表信息.**缓存**到本地

1. 当消费者调用服务时,查找缓存到本地的服务列表信息.之后通过负载均衡机制挑选其中一个服务进行访问.
2. 当后台服务器宕机时,由于注册中心有心跳检测机制.所以可以发现某台服务器宕机.之后更新自己的服务列表信息.之后广播给全部消费者.
3. 当消费者获取服务列表的通知时,最终更新本地的服务列表数据.

## Zookeeper注册中心

### Zookeeper介绍

ZooKeeper是一个[分布式](https://baike.baidu.com/item/%E5%88%86%E5%B8%83%E5%BC%8F/19276232)的，开放源码的[**分布式应用程序**](https://baike.baidu.com/item/%E5%88%86%E5%B8%83%E5%BC%8F%E5%BA%94%E7%94%A8%E7%A8%8B%E5%BA%8F/9854429)**协调服务**，是[Google](https://baike.baidu.com/item/Google)的Chubby一个[开源](https://baike.baidu.com/item/%E5%BC%80%E6%BA%90/246339)的实现，是Hadoop和Hbase的重要组件。**它是一个为分布式应用提供一致性服务的软件**，提供的功能包括：配置维护、域名服务、分布式同步、组服务等。

ZooKeeper的目标就是封装好复杂易出错的关键服务，将简单易用的接口和性能高效、功能稳定的系统提供给用户。

ZooKeeper包含一个简单的原语集，[1]  提供Java和C的接口。

ZooKeeper代码版本中，提供了分布式独享锁、选举、队列的接口，代码在zookeeper-3.4.3\src\recipes。其中分布锁和队列有[Java](https://baike.baidu.com/item/Java/85979)和C两个版本，选举只有Java版本。（概述图片来源：[2]  ）

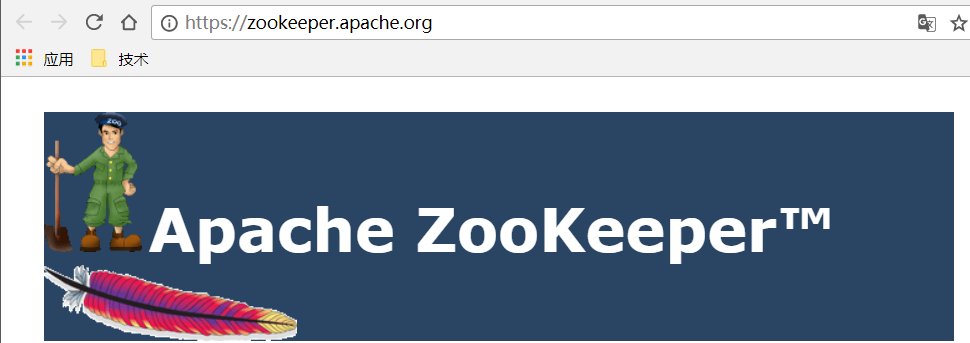
总结:zookeeper是服务的协调调度服务器!!!

## Zookeeper安装使用

### Zookeeper下载

网址: <http://zookeeper.apache.org/releases.html>.

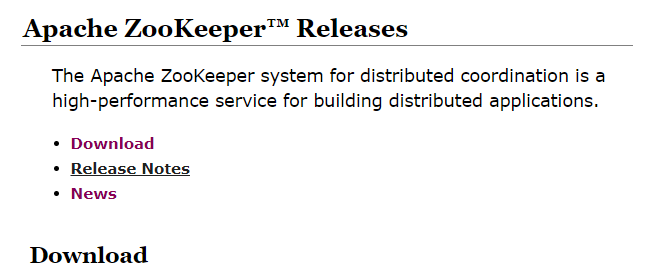
如图-2所示



**图-**2

下载路径,点击download.

如图-3所示

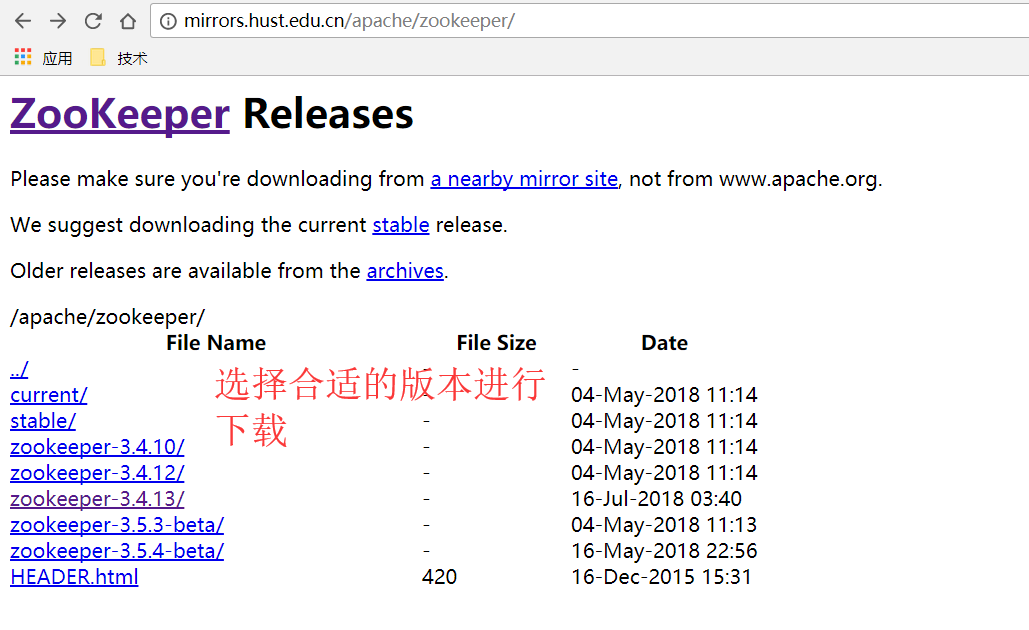


**图-**3

下载Zookeeper地址.

<http://mirrors.hust.edu.cn/apache/zookeeper/>

如图-4所示



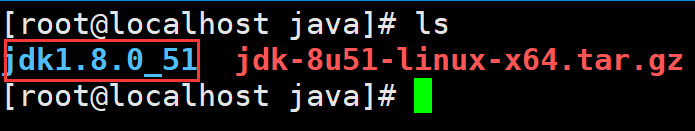
**图-**4

## Zookeeper安装

### 安装JDK

将JDK1.8文件上传到Linux操作系统中/src/usr/local/java/文件下.

如图-5所示



**图-**5

1.解压文件

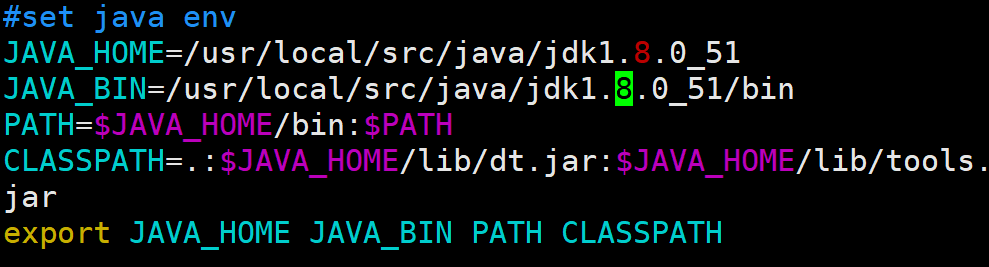
tar -xvf jdk-8u51-linux-x64.tar.gz

2.配置环境变量

编辑环境变量配置文件

vim /etc/profile

如图-6所示

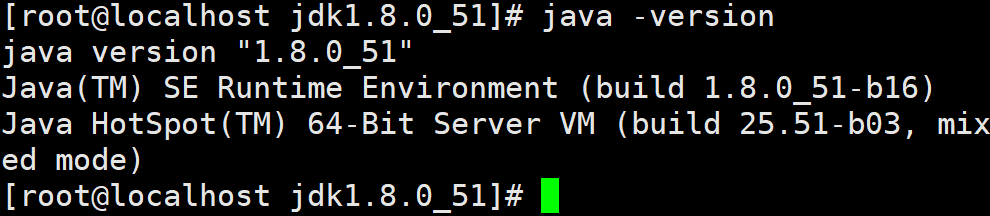


**图-**6

使JDK生效,之后检查jdk安装是否成功

source /etc/profile

如图 -7所示

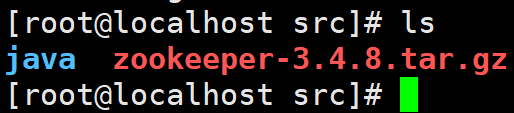


**图-**7

### 上传安装文件

说明:上传zookeeper安装文件.之后解压.

如图-8所示



**图-**8

解压目录:

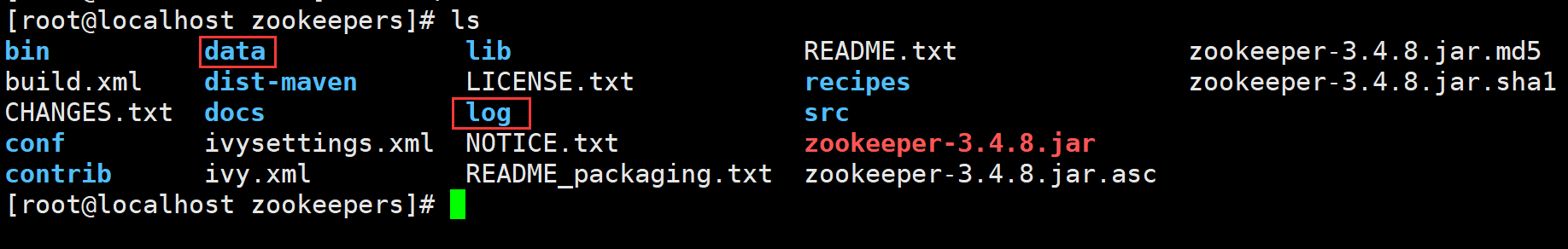
tar -xvf zookeeper-3.4.8.tar.gz

### 修改配置文件

在zk根目录下创建文件夹data/log

mkdir data log

如图-9所示

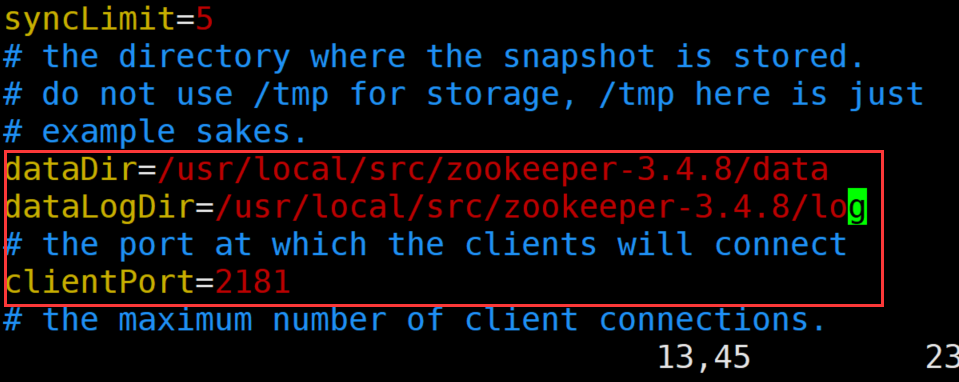


**图-**9

复制配置文件并且修改名称

cp zoo\_sample.cfg zoo.cfg

如图-10所示



**图-**10

### 启动zk

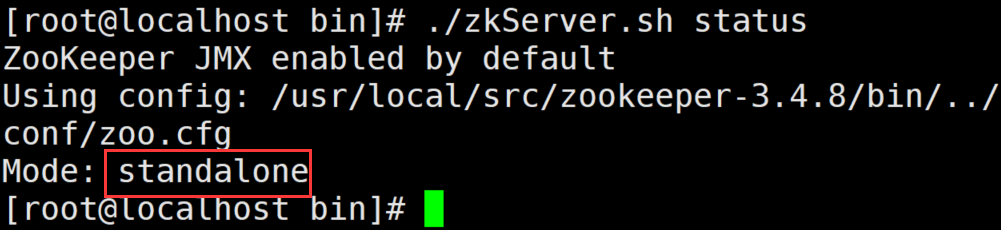
zk启动关闭命令如下.

sh zkServer.sh start 或者 ./zkServer.sh start

sh zkServer.sh stop

sh zkServer.sh status

如图-11所示



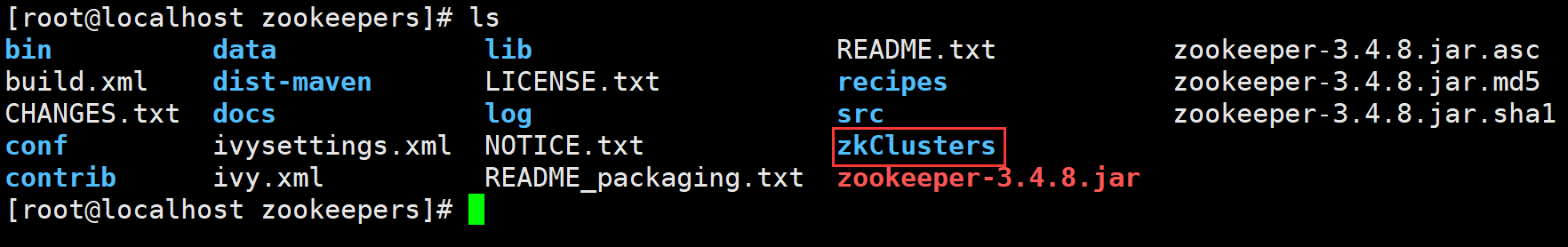
**图-**11

## Zookeeper集群安装

### 准备文件夹

在zookeeper根目录中创建新的文件夹zkCluster.

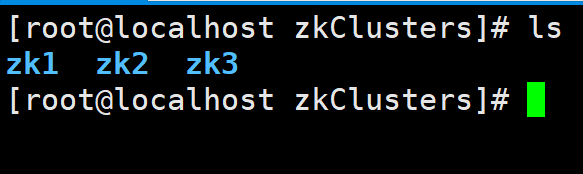
如图-12所示



**图-**12

如图-13所示

创建zk1/zk2/zk3文件夹.

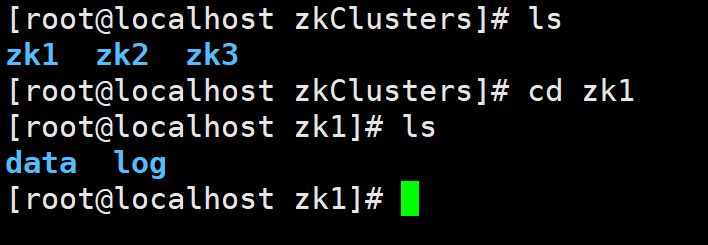


**图-**13

在每个文件夹里创建data/log文件夹.

mkdir {zk1,zk2,zk3}/{data,log}

如图-14所示

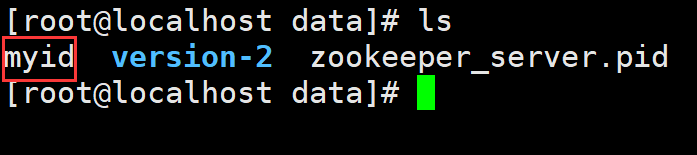


**图-**14

### 添加myid文件

分别在zk1/zk2/zk3中的data文件夹中创建新的文件myid.其中的内容依次为1/2/3,与zk节点号对应.

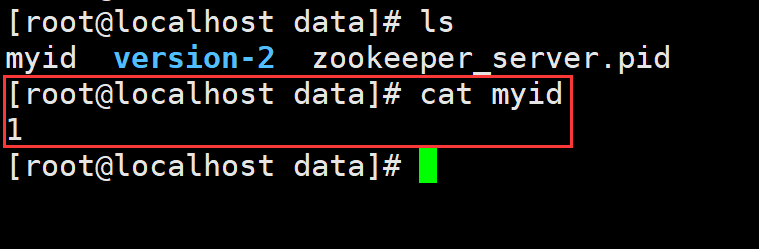
如图-15所示



**图-**15

编辑myid文件,定义编号.

如图-16所示

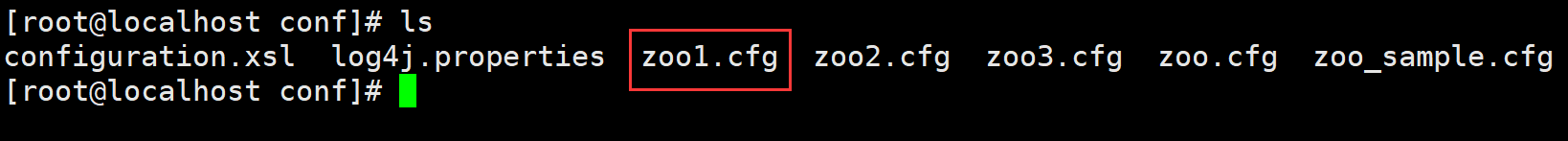


**图-**16

### 编辑配置文件

将zoo\_sample.cfg 复制为zoo1.cfg之后修改配置文件.

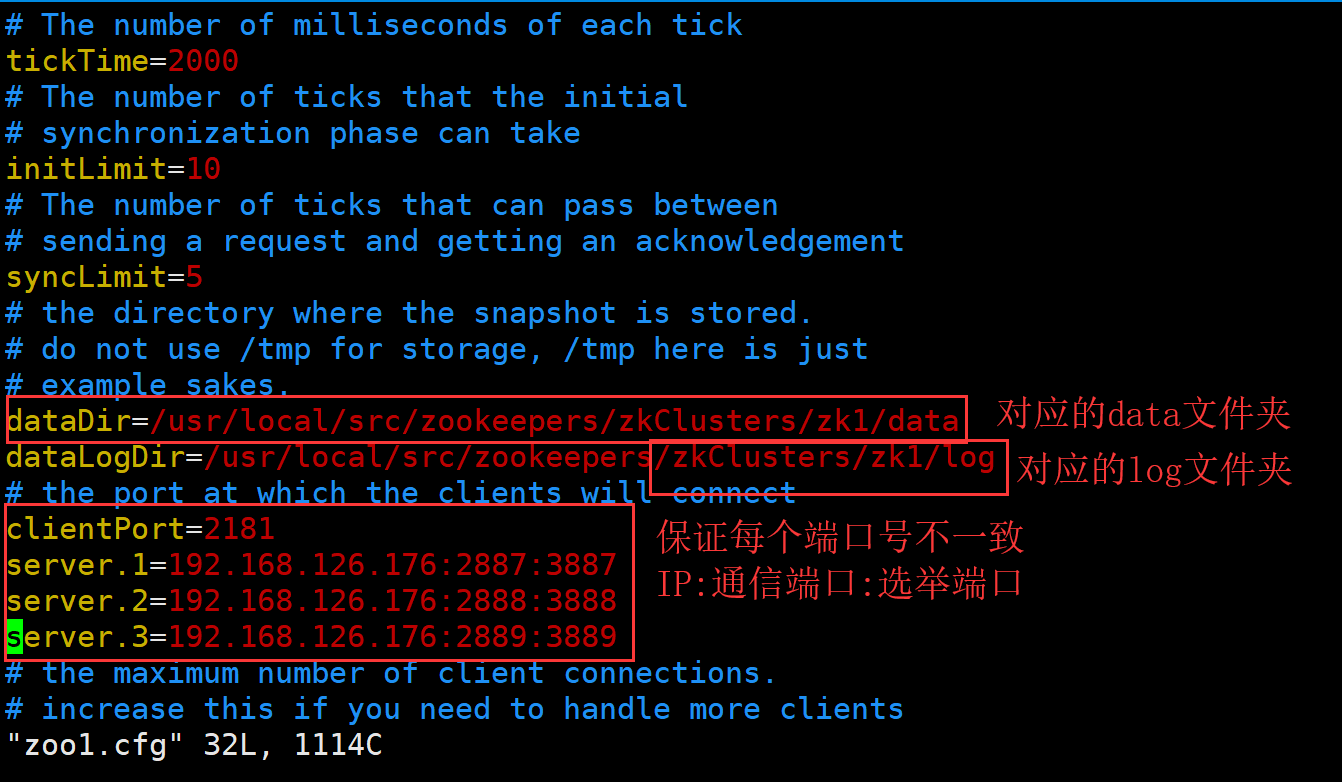
如图-17所示



**图-**17

### 修改zoo1.cfg

如图-18所示



**图-**18

配置完成后将zoo1.cfg复制2份.之后需要修改对应的文件夹目录.和不同的端口即可.

### ZK集群测试

通过下面的命令启动zk集群.

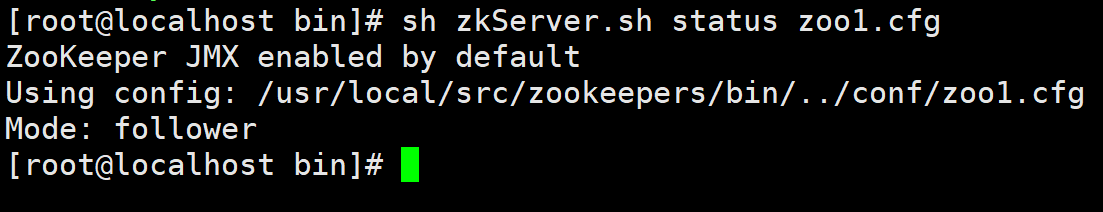
sh zkServer.sh start zoo1.cfg

sh zkServer.sh stop zoo1.cfg

sh zkServer.sh status zoo1.cfg

检查主从关系,从机情况说明.

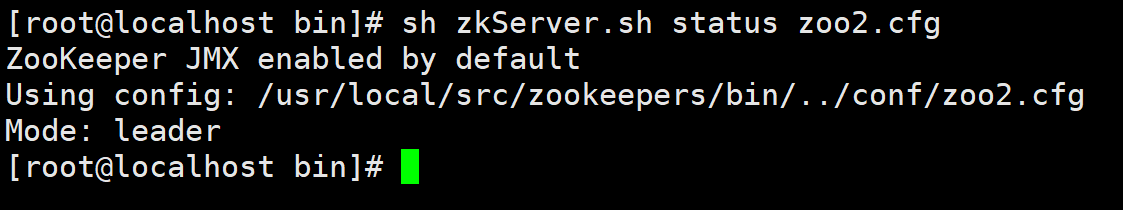
如图-19所示



**图-**19

检查主从关系,主机情况说明.

如图-20所示



**图-**20

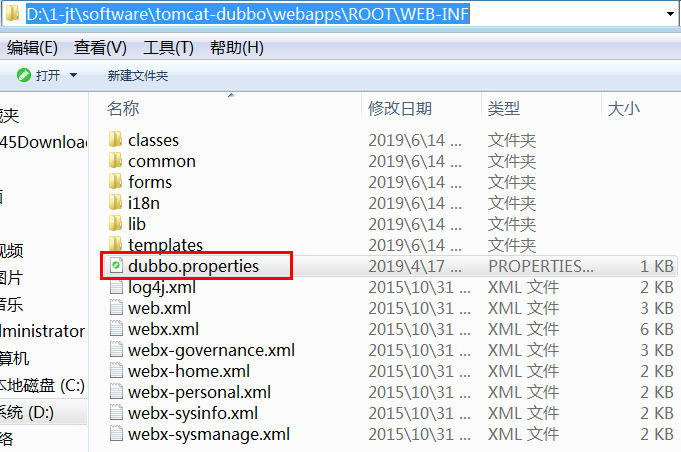
### 关于zookeeper集群说明

Zookeeper集群中leader负责监控集群状态,follower主要负责客户端链接获取服务列表信息.同时参与投票.

**图-**30

### 搭建dubbo控制台

说明:控制台是监控服务的.修改dubbo的配置文件信息.改为zkIP地址和端口



2.修改配置文件信息

dubbo.registry.address=zookeeper://192.168.175.129:2181?backup=192.168.175.129:2182,192.168.175.129:2183

dubbo.admin.root.password=root

dubbo.admin.guest.password=guest

3.控制台展现

用户名和密码都是root 登录之后进入控制台.



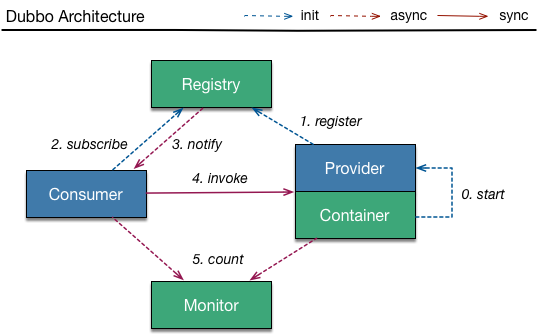
## Dubbo框架

### 介绍

Dubbo是[1]  阿里巴巴公司开源的一个高性能优秀的[服务框架](https://baike.baidu.com/item/%E6%9C%8D%E5%8A%A1%E6%A1%86%E6%9E%B6" \t "_blank)，使得应用可通过高性能的 RPC 实现服务的输出和输入功能，可以和 [2]  [Spring](https://baike.baidu.com/item/Spring)框架无缝集成。

Dubbo是一款高性能、轻量级的开源Java RPC框架，它提供了三大核心能力：面向接口的远程方法调用，智能容错和负载均衡，以及服务自动注册和发现。

### Dubbo工作原理



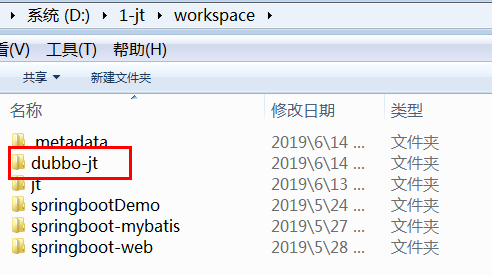
1. consumer 服务消费者
2. provider 服务提供者
3. registry 注册中心

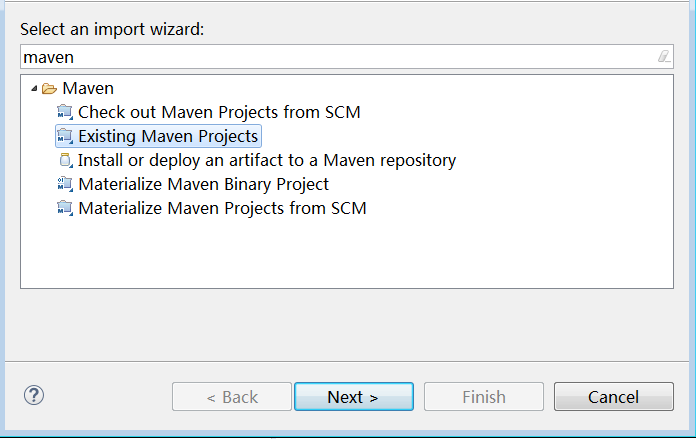
基于SOA思想

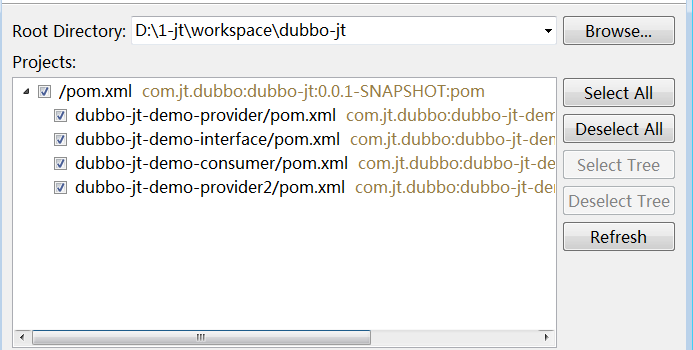
## Dubbo入门案例

### 导入项目

说明:将课前资料中的dubbo项目导入工作空间

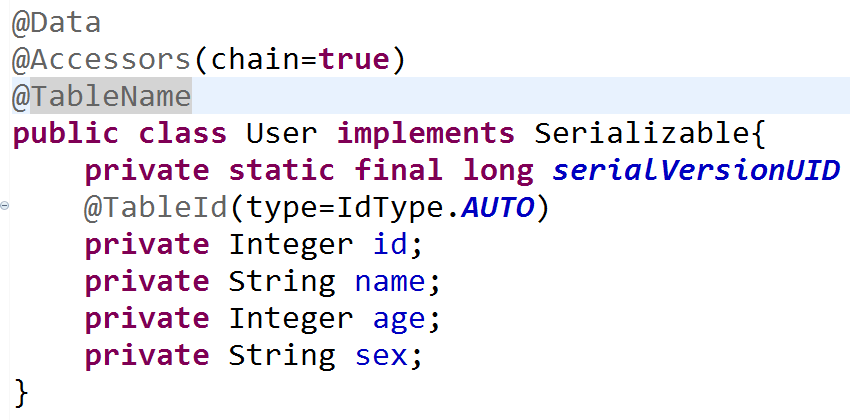






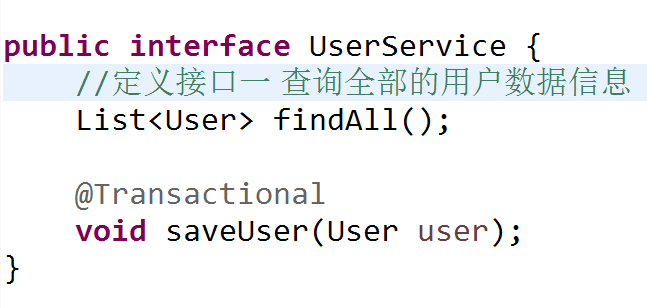
### 定义pojo对象

说明:在jt-interface中定义pojo对象



### 定义UserService接口

说明:基于SOA思想系统与系统直接通信,需要通过接口进行调用.



### 编辑提供者

#### 定义实现类

说明:利用@Service注解标识提供者.

@Service(timeout=3000) //3秒超时

**public** **class** UserServiceImpl **implements** UserService {

@Autowired

**private** UserMapper userMapper;

@Override

**public** List<User> findAll() {

System.***out***.println("我是第一个服务的提供者");

**return** userMapper.selectList(**null**);

}

//user对象是消费者利用rpc通信传递

@Override

**public** **void** saveUser(User user) {

userMapper.insert(user);

}

}

### 编辑提供者配置文件

server:

port: 9000

spring:

datasource:

#引入druid数据源

type: com.alibaba.druid.pool.DruidDataSource

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

url: jdbc:mysql://127.0.0.1:3306/jtdb?serverTimezone=GMT%2B8&useUnicode=true&characterEncoding=utf8&autoReconnect=true&allowMultiQueries=true

username: root

password: root

#Springboot整合dubbo

dubbo:

scan:

basePackages: com.jt #包扫描dubbo的service注解

application:

name: provider-user #提供者服务名称

registry: #链接zookeeper

address: zookeeper://192.168.175.129:2181?backup=192.168.175.129:2182,192.168.175.129:2183

protocol:

name: dubbo

port: 20880 #服务通信端口 消费者链接提供者端口

mybatis-plus:

type-aliases-package: com.jt.dubbo.pojo #配置别名包路径

mapper-locations: classpath:/mybatis/mappers/\*.xml #添加mapper映射文件

configuration:

map-underscore-to-camel-case: true #开启驼峰映射规则

### 控制台监控

当启动服务提供者时.检查控制台启动是否正常.



## 编辑消费者

### 编辑Controller

@RestController

**public** **class** UserController {

/\*\*

\* springboot利用dubbo注解实例化接口对象

\* timeout=3000 定义超时时间

\* check=true 当服务器启动时检查是否有提供者

\* 如果没有提供者,则程序启动报错.

\* check=false

\* 当程序启动时不会检查是否有提供者.当程序调用

\* 时检查.

\*/

@Reference(timeout=3000,check=**false**)

**private** UserService userService;

@RequestMapping("/findAll")

**public** List<User> findAll(){

**return** userService.findAll();

}

@RequestMapping("/saveUser/{name}/{age}/{sex}")

**public** String saveUser(User user) {

userService.saveUser(user);

**return** "用户入库成功!!!";

}

}

### 编辑消费者yml配置

server:

port: 9001

dubbo:

scan:

basePackages: com.jt

application:

name: consumer-user #服务名称"唯一"

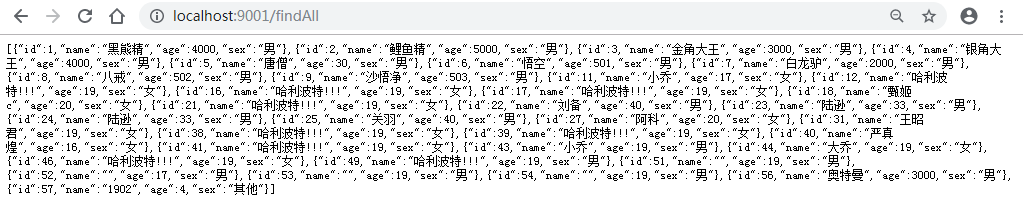
registry:

address: zookeeper://192.168.175.129:2181?backup=192.168.175.129:2182,192.168.175.129:2183

### 控制台监控



### 消费者测试



### 作业

1. 将dubbo的入门案例 自己手写.
2. 完成user的CURD操作.
3. 将dubbo的调用流程 熟记.