# ======知识点======

## 什么是Zookeeper？

Zookeeper是一个高效的分布式协调服务，它暴露了一些公共服务，比如命名/配置/管理/同步控制/群组服务等。我们可以使用ZK来实现比如达成共识/集群管理/leader选举等。

**Zookeeper是一个高可用的分布式管理与协调框架，基于ZAB算法（原子消息广播协议）的实现**。该框架能够很好地保证分布式环境中数据的一致性。也正是基于这样的特性，使得Zookeeper成为了解决分布式一致性问题的利器。

### 特性

1. **顺序一致性：**

从一个客户端发起的事物请求，最终将会严格地按照其发起的顺序被应用到Zookeeper中去。

1. **原子性：**

所有事物请求的处理结果在整个集群中所有机器上的应用情况是一致的，也就是说，要么整个集群所有的机器都成功应用了某一事物，要么就没有应用，一定不会出现部分机器应用了该事物，而另一部分没有应用的情况。

1. **单一视图：**

无论客户端连接的是哪一个zookeeper服务器，其看到的服务器端数据模型都是一致的。

1. **可靠性：**

一旦服务器成功地应用了一个事物，并完成了对客户端的响应，那么该事物所引起的服务器端状态将会被一致保留下来。除非有另一个事物对其更改。

1. **实时性：**

通常所说的实时性就是指一旦事物被成功应用，那么客户端就能立刻从服务器上获取变更后的新数据，zookeeper仅仅能保证在一段时间内，客户端最终一定能从服务器端读取最新的数据状态。

|  |
| --- |
| 举例说明上述特性：  3个节点通过paxos算法，ZAB协议选举出1个leader，2个follower，假如leader挂了，只要半数以上节点是好的就能对外提供服务。  IMG_256  顺序一致性、原子性：三个节点构成集群，相当于一个整体，如果有一个client端发起一个请求把某个节点的数据1改为2，zk会把通过ZAB把修改的消息发给其他节点，用的同步算法就是paxos算法。同步数据期间不允许其他client修改数据。  可靠性：事物成功后会给client反馈，一问一答的形式 |

### Zookeeper设计目标

**目标1：简单的数据结构。**

zookeeper就是以简单的树形结构来进行相互协调的（也叫树形命名空间）

**目标2：可以构建集群。**

一般zookeeper集群通常由一组机器构成，一般3~5台机器就可以组成一个zookeeper集群了。只要集群中超过半数以上的机器能够正常工作，那么整个集群就能够正常对外提供服务。

**目标3：顺序访问。**

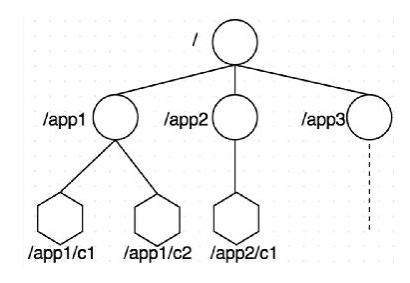
对于来自每一个客户端的每一个请求，zookeeper都会分配一个全局唯一的递增编号，这个编号反应了所有事物操作的先后顺序，应用程序可以使用zookeeper的这个特性来实现更高层次的同步。

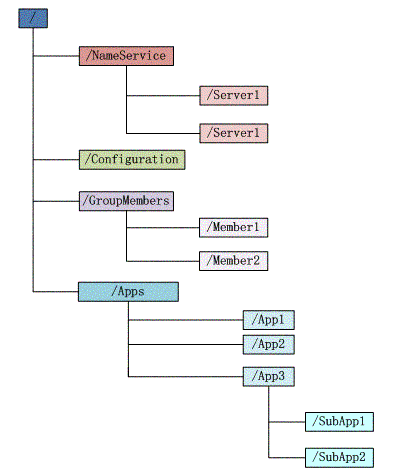
**目标4：高性能。**

由于zookeeper将全量数据存储在**内存中**，并直接服务于所有的非事物请求，因此尤其在读操作为主的场景下性能非常突出。

### Zookeeper数据结构

Zookeeper会维护一个具有层次关系的数据结构，它非常类似于一个标准的文件系统





### Zookeeper数据模型

1，每个子目录项如 NameService 都被称作znode，这个znode是被它所在的路径唯一标识，如Server1这个 znode的标识为 /NameService/Server1

2，znode 可以有子节点目录，并且每个znode可以存储数据，注意EPHEMERAL类型的目录节点不能有子节点目录

3，znode是有版本的，每个znode中存储的数据可以有多个版本，也就是一个访问路径可以存储多份数据

4，znode可以是临时节点，一旦创建这个znode的客户端与服务器失去联系，这个znode也将自动删除，Zookeeper的客户端和服务器通信采用长连接方式，每个客户端和服务器通过心跳来保持连接，这个连接状态称为session，如果znode是临时节点，这个session失效，znode也就删除了。

5，znode 的目录名可以自动编号，如App1 已经存在，再创建的话，将会自动命名为App2

6，znode 可以被监控，包括这个目录节点中存储的数据的修改，子节点目录的变化等，一旦变化可以通知设置监控的客户端，这个是Zookeeper的核心特性，Zookeeper的很多功能都是基于这个特性实现的

### Zookeeper组成

ZK Server根据其身份特性分为三种，Leader ，Follower，Observer（就是一个client端），其中Follower和Observer又统称为Learner（学习者）。

　　Leader：负责客户端的writer 类型请求

　　Follower：负责客户端的reader 类型请求，参与leader选举等。

Observer：特殊的“Follower“，其可以接受客户端reader 请求，但不参与选举。（扩容系统支撑能力，提高了读取速度。因为它不接受任何同步的写入请求，只负责与leader同步数据）

### Zookeeper典型应用场景

Zookeeper 从设计模式的角度来看，是一个基于【**观察者模式**】**设计的分布式服务管理框架，**它负责存储和管理大家都关心的数据，然后接受观察者的注册，一旦这些数据的状态发生变化，Zookeeper 就将负责通知已经在Zookeeper 上注册的那些观察者做出相应的反应，从而实现集群中类似 Master/Slave 管理模式。

1，配置管理

2，集群管理

3，发布与订阅

4，数据库切换

5，分布式日志的收集

6，分布式锁、队列管理等

**应用场景说明**

**一：配置管理**

配置的管理在分布式应用环境中很常见，比如我们在平常的应用系统中，经常会遇到这样的需求，如机器的配置列表、运行时的开关配置、数据库配置信息等。这些全局配置信息通常具有以下3个特性：

1，数据量比较小

2，数据内容在运行时动态发生变化

3，集群中各个节点共享信息，配置一致

**二：集群管理**

Zookeeper 不仅能够帮你维护当前的集群中机器的服务状态，而且能够帮你选出一个“总管” ，让这个总管来管理集群，这就是 Zookeeper 的另一个功能 Leader，并实现集群容错功能。

1，希望知道当前集群中究竟有多少机器工作

2，对集群中每天集群的运行状态进行数据收集

3，对集群中每台集群进行上下线操作

**三：发布与订阅，**

Zookeeper 是一个典型的发布/订阅模式的分布式数控管理与协调框架，开发人员可以使用它来进行分布式数据的发布与订阅

**四：数据库切换，**

比如我们初始化Zookeeper 的时候读取节点上的数据库配置文件，当配置一旦发生变更时，zookeeper 就能帮我们把变更的通知发送到各个客户端，每个客户端在接收到这个变更后，就可以从新进行最新数据的获取。（数据库properties配置在zk，挂了的话切换到另一个）

五：分布式日志收集，我们可以做一个日志系统收集集群中所有的日志信息，进行统一管理。

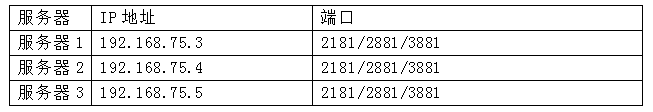
六：Zookeeper 的特性就是在分布式场景下高可用，但是用原生的API 实现的分布式功能非常困难，可以使用第三方客户端的完美实现，比如Curator框架，它是 Apache 的顶级项目

### Zookeeper开源框架的应用

如 Hadoop、Storm、消息中间件、RPC服务框架、数据库增量订阅与消费组件（如mysql binlog）、分布式数据库同步系统，淘宝的Otter。

## ZK集群安装

**1，准备三台机器做集群，都已安装JDK**



1. **下载**

下载zookeeper-3.4.10.tar.gz 到/data/program/software/目录，

wget <http://mirror.bit.edu.cn/apache/zookeeper/zookeeper-3.4.9/zookeeper-3.4.9.tar.gz>

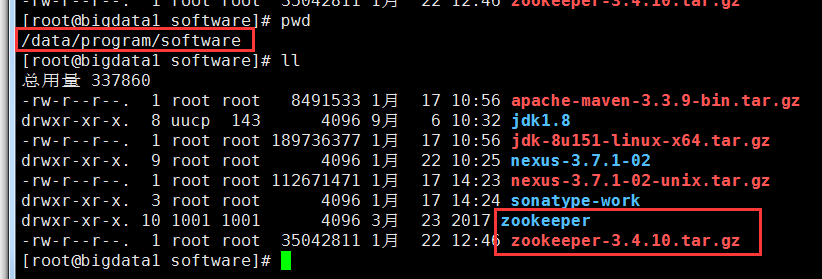
**3，解压zookeeper安装包，并对节点重命名**

#tar -zxvf zookeeper-3.4.10.tar.gz

服务器1：#mv zookeeper-3.4.10 zookeeper

服务器2：#mv zookeeper-3.4.10 zookeeper

服务器3：#mv zookeeper-3.4.10 zookeeper

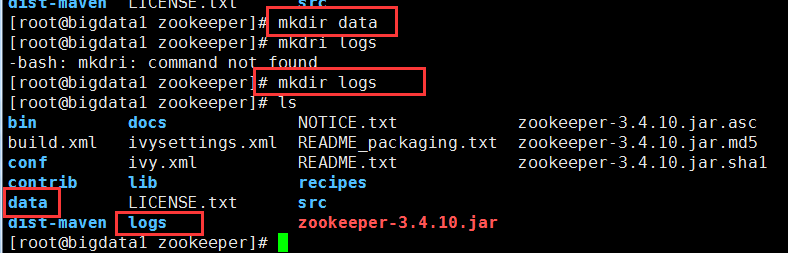


1. **在zookeeper的各个节点下 创建数据和日志目录**

#cd zookeeper

#mkdir data

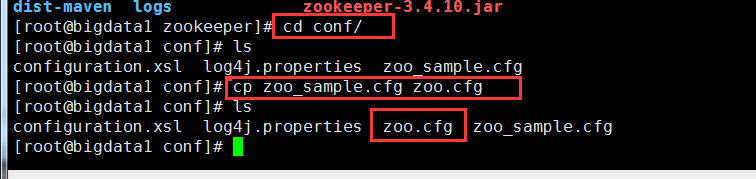
#mkdir logs



1. **配置zoo.cfg**

将zookeeper/conf目录下的zoo\_sample.cfg文件拷贝一份，命名为zoo.cfg

#cp zoo\_sample.cfg   zoo.cfg



vim 修改zoo.cfg 配置文件，修改如下几处，保存

　　　　　　clientPort=2181

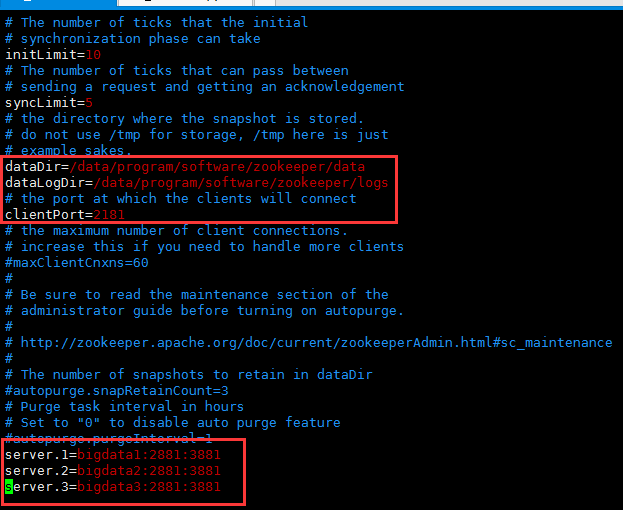
　　　　　　dataDir=/data/program/software/zookeeper/data

　　　　　　dataLogDir=/data/program/software/zookeeper/logs

　　　　　　server.1=bigdata1:2881:3881

　　　　　　server.2=bigdata2:2881:3881

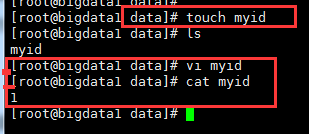
　　　　　　server.3=bigdata3:2881:3881



|  |
| --- |
| zoo.cfg说明：  [root@bigdata1 conf]# cat zoo.cfg  tickTime=2000  initLimit=10  syncLimit=5  dataDir=/data/program/software/zookeeper/data  dataLogDir=/data/program/software/zookeeper/logs  clientPort=2181  server.1=bigdata1:2881:3881  server.2=bigdata2:2881:3881  server.3=bigdata3:2881:3881  参数说明：  **tickTime=2000** 　　tickTime这个时间是作为Zookeeper服务器之间或客户端与服务器之间维持心跳的时间间隔,也就是每个tickTime时间就会发送一个心跳。  **initLimit=10** 　　initLimit这个配置项是用来配置Zookeeper接受客户端（这里所说的客户端不是用户连接Zookeeper服务器的客户端,而是Zookeeper服务器集群中连接到Leader的Follower 服务器）初始化连接时最长能忍受多少个心跳时间间隔数。当已经超过10个心跳的时间（也就是tickTime）长度后Zookeeper 服务器还没有收到客户端的返回信息,那么表明这个客户端连接失败。总的时间长度就是10\*2000=20 秒。  **syncLimit=5** 　　syncLimit这个配置项标识Leader与Follower之间发送消息,请求和应答时间长度,最长不能超过多少个tickTime的时间长度,总的时间长度就是5\*2000=10秒。  **dataDir=/data/program/software/zookeeper/data** 　　dataDir顾名思义就是Zookeeper保存数据的目录,默认情况下Zookeeper将写数据的日志文件也保存在这个目录里。  **clientPort=2181** 　　clientPort这个端口就是客户端（应用程序）连接Zookeeper服务器的端口,Zookeeper会监听这个端口接受客户端的访问请求。  **server.A=B：C：D** 　　server.1=bigdata1:2881:3881 　　server.2=bigdata2:2881:3881 　　server.3=bigdata3:2881:3881  　　A是一个数字,表示这个是第几号服务器； 　　B是这个服务器的IP地址（或者是与IP地址做了映射的主机名）； 　　C此端口表示，用来集群成员的信息交换，表示这个服务器与集群中的Leader服务器交换信息的端口； 　**D是在leader挂掉时专门用来进行选举leader所用的端口。** 　　注意：如果是伪集群的配置方式，不同的 Zookeeper 实例通信端口号不能一样，所以要给它们分配不同的端口号。 |

1. **创建myid文件**

在 dataDir=/data/program/software/zookeeper/data 下创建myid文件



编辑myid文件，并在对应的IP的机器上输入对应的编号。如在1上，myid文件内容就是1, 2上就是2， 3上就是3：   
　　bigdata1    #vi /myid## 值为1  
　　bigdata2    #vi /myid## 值为2  
　　bigdata3    #vi /myid## 值为3

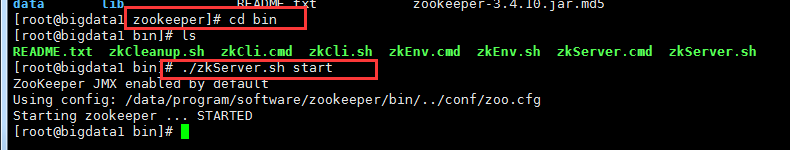
bigdata1 192.168.75.3 的配置就配置完了，bigdata2 、bigdata3也照着配置。

7**启动测试zookeeper**

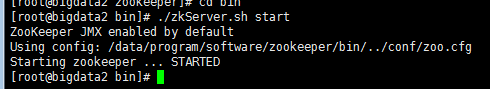
(1)进入/bin目录下执行：

　　　# ./zkServer.sh  start

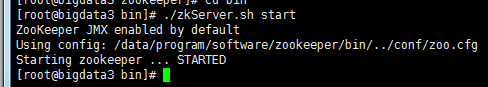
　　　bigdata1:



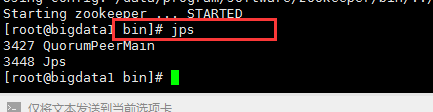
　bggdata2:



bigdata3:



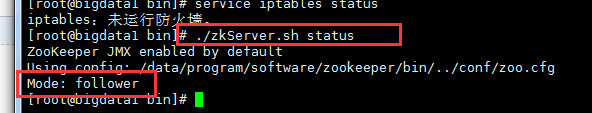
输入jps命令查看进程：



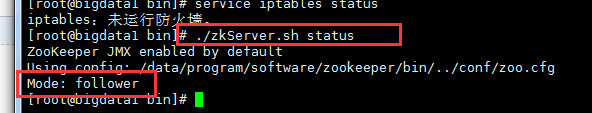
其中，QuorumPeerMain是zookeeper进程，说明启动正常

查看状态：（需关闭防火墙）

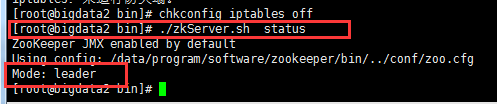
# ./zkServer.sh status



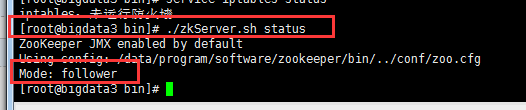
　bigdata1: 是一个follower



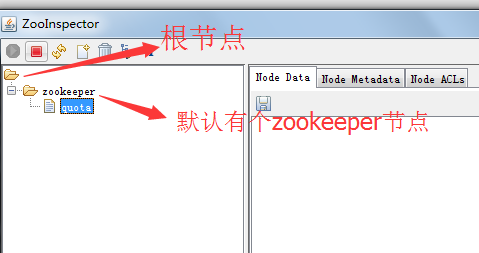
bigdata2:



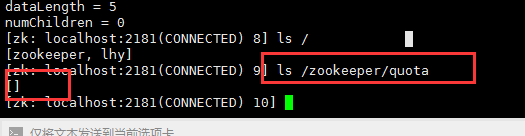
bigdata3:



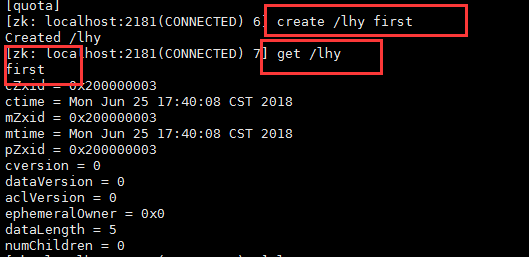
## ZK常用命令



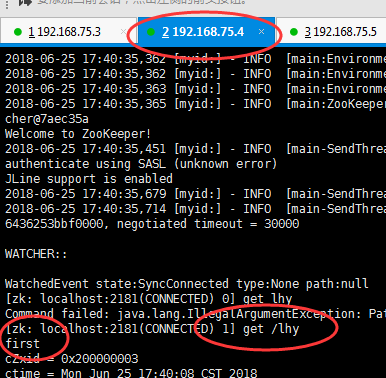
**查找 ： ls  /      ls /zookeeper**



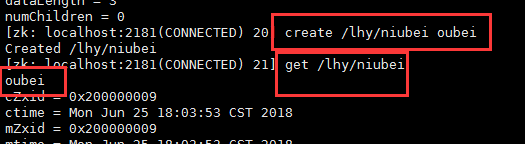
**创建并赋值：create  /lhy first**

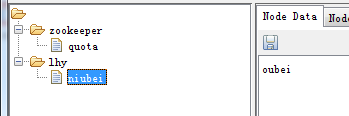


**创建是在192.168.75.3节点上创建的，其他节点也会有数据**

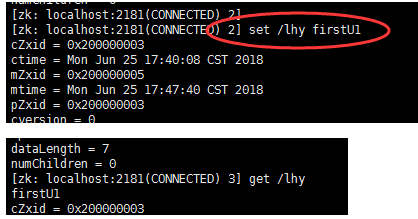


创建子节点





**设值新值覆盖旧值：set  /lhy second**



**递归删除 rmr  /path**

**删除指定的某个节点 delete  /path/child**

