# Zookeeper（动物园铲屎官）

## 安装：

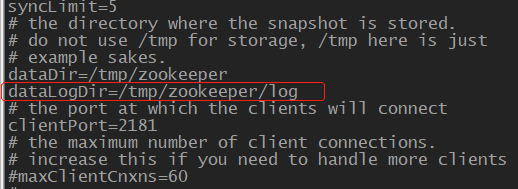
下载地址：<http://mirror.bit.edu.cn/apache/zookeeper/>（不要下载不稳定的版本，会启动成功，但是查看状态会显示失败，我是下过3.5.5版本的就出现过这个情况，网上各种答案都不行，后面直接换了3.4.14的直接OK）

还有就是非正常关闭的，再次启动有时候会失败，这个时候因为在log文件夹下（con.cfg配置的DataDir=xxx）会有一个zookeeper\_server.pid的文件，里面记录了上次启动的ID,如果启动失败，可以尝试将这个删除了，然后重新启动



解压: tar -xvf xxx.tar

配置文件：复制一份conf 下的zoo\_sample.cfg，命名为zoo.cfg，多添加一项配置，dataLogDir=/tem/zookeeper/log 这个路径是要自己创建的，为了方便，我就直接在原本的文件下面创建了一个log文件夹



zoo.cfg的关键配置有3个：

tickTime=2000：一次心跳的基本时间，

dataDir：数据与日志的存放处(如果没有文件夹会报错，自己手动创建)

clientPort：端口号(默认2181)

环境变量的配置：

vi /etc/profile 打开环境变量配置

添加如下选项：

export ZOOKEEPER=/usr/local/zookeeper（安装路径）

export PATH=$PATH:$ZOOKEEPER/bin

source /etc/profile //使生效

配置了环境变量之后，就可以不在bin目录下也直接调用命令启动

## Zookeeper的简介

zookeeper就是一种用于分布式应用程序的高性能协调服务，它的特点就是数据是存于内存中的，持久化实现在日志中。它的内存类似于树形结构，且搞吞吐量低延迟，可以帮助我们实现分布式统一配置中心，服务注册，分布式锁等。组成ZooKeeper服务的服务器必须彼此了解。他们维护内存中的状态图形，以及持久性存储中的事务日志和快照。只要大多数服务器可用，ZooKeeper服务就可用。客户端连接到单个ZooKeeper服务器。客户端维护TCP连接，通过该连接发送请求，获取响应，获取监视时间以及发送tick。如果与服务器的TCP连接中断，则客户端将连接到其它服务器。

## Zookeeper的特点

### 1.数据结构简单

类似于Unix文件系统树形结构，每个目录成为Znode节点，但它不同于文件系统，它既可以视为文件夹，也可以视为文件来存放数据，但是我们平时还是得叫它节点，别叫文件夹这么掉价。

需要注意：同一个节点下的儿子节点名称不能相同，且命名是有规范的，它的路径是没有相对路径的概念的，都是绝对路径，任何开始都以“/”开始，且起存储数据的大小是有限制的

### 2.数据模型特点

层次命名空间：就是上面已经提到的，类似于unix的文件系统，以"/"为根，节点可以包含关联数据和子节点，绝对路径 Znode：名称唯一，命名有规范，类型分4种：持久，顺序，临时，临时顺序，节点的数据构成之后再提

### 3.有序性

提供多种方式跟踪时间，ZooKeeper给每个更新贴上一个数字，这个数字反映了所有ZooKeeper事务的顺序，严格的顺序意味着可以在客户机上实现复杂的同步原语 解释czxid、version、zoo.cfg中ticks配置

### 4.特性

* Session会话

1. 一个客户端连接一个会话，由Zookeeper分配唯一会话ID
2. 客户端以特定的时间间隔发送心跳以保持会话有效性
3. 超过会话间隔时间未收到客户端的心跳，则判断客户端无效（默认2倍ticktime）
4. 会话中的请求是FIFO（先进先出）的顺序执行

* Znode的数据构成

节点数据：存储的基本信息（状态，配置，位置等）

节点元数据：stat命令下的一些数据

数据大小：限制1M

* Znode的节点类型

1. 持久节点：直接通过create path value所创建
2. 临时节点：create -e path value
3. 顺序节点：create -s path value

注意：

1. Session会话失效时，临时节点就会被删除
2. 顺序节点的创建，后10位十进制序号，每个父节点拥有一个技术器，这个计数器也是有限制的，到2147483647
3. 顺序节点会话结束仍然存在

* Watch监听机制

客户端能在Znode上设置watch,监听Znode的变化，包括增删改查，通过stat path,ls2 path,get path皆可查看

触发wath事件的条件有4种，create,delete,change,child（子节点事件）

* Watch的重要特性

1. 仅一次性：watch触发后立即删除，要持续监听变化的话就要持续提供设置watch,这也是watch的注意事项
2. 有序性：客户端先得到watch通知才可查看变化结果

* Watch的注意事项

1. 刚刚提及的它的仅一次性
2. 获取事件和发送watch,获取watch,这些请求可能存在延时，所以不能绝对可靠的得到每个节点发生的更改
3. 一个watch对象只会被通知一次，如果一个watch同时注册了多个接口（exists,getData）,如果此时删除节点，虽然这个事件对exists和getData都有效，但是watch只会被调用一次

## Zookeeper的特性

1. 顺序一致性，保证客户端操作都是按顺序生效的
2. 原子性，更新成功或者失败，没有部分结果
3. 单个系统映象，无论连接到哪个服务器，客户端都将看到相同的内容
4. 可靠性，数据的变更不会丢失，除非被客户端覆盖修改
5. 及时性，保证系统的客户端当时读取到的数据是最新的

## Zookeeper 实现分布式锁

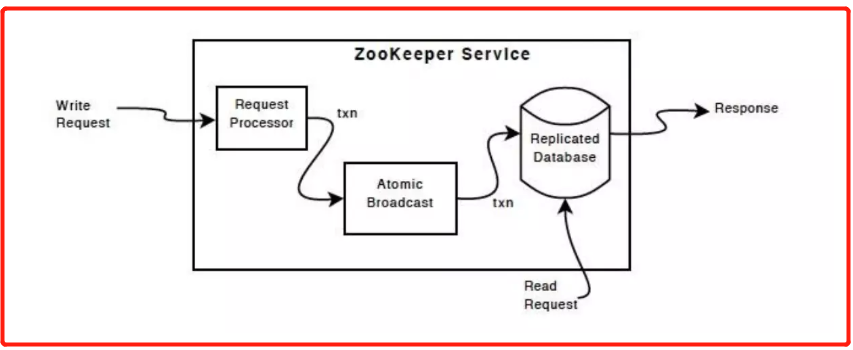
特点：

1. Zookeeper 由于搭建集群比较容易，所以可以防止因为单点系统而导致数据丢失的问题。
2. Zookeeper 的有序性，原子性。可以保证每次线程的操作都是有顺序的。
3. 节点的临时性：不必担心死锁问题，有心跳检测，过期的临时节点可以自动删除，然后释放锁

## zookeeper集群的工作原理

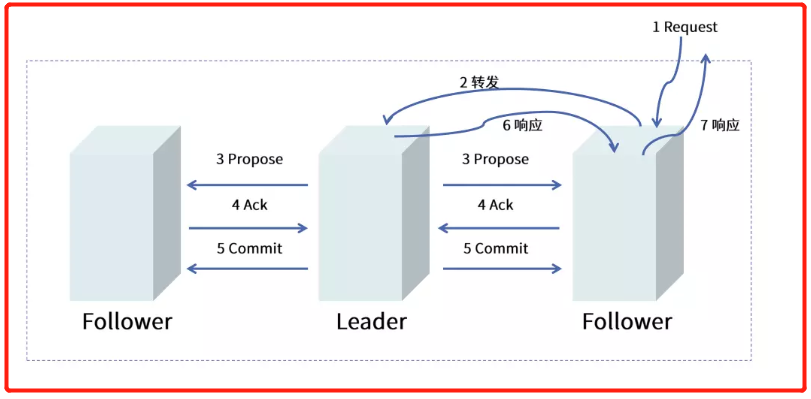
zookeeper提供了重要的分布式协调服务，它是如何保证集群数据的一致性的？

### ZAB协议的简单描述



ZAB（zookeeper atomic broadcast）---zookeeper 原子消息广播协议是专门为zookeeper设计的数据一致性协议，注意此协议最主要的关注点在于数据一致性，而无关乎于数据的准确性，权威性，实时性。

ZAB协议过程



1.所有事务转发给leader（当我们的follower接收到事务请求）

2.Leader分配全局单调递增事务id（zxid，也就是类似于paxos算法的编号n），广播协议提议

3.Follower处理提议，作出反馈（也就是承诺只接受比现在的n编号大的

4.leader收到过半数的反馈，广播commit，把数据彻底持久化（和2pc不同的是，2pc是要等待所有小弟反馈同意）

5.leader对原来转发事务的followe进行响应，follower也顺带把响应返回给客户端

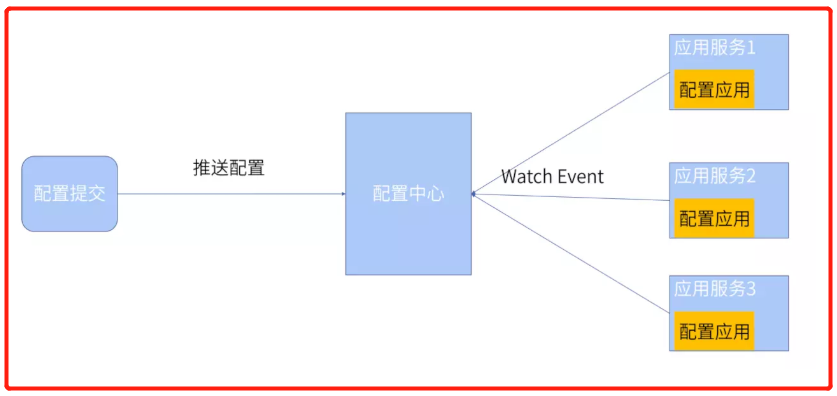
还记得我们说过zookeeper比较适合读比较多，写比较少的场景吗，为什么我们说它效率高，我们可以知道，所有的事务请求，必须由一个全局唯一的服务器进行协调，这个服务器也就是现在的leader，leader服务器把客户端的一个写请求事务变成一个提议，这个提议通过我们的原子广播协议广播到我们服务器的其他节点上去，此时这个协议的编号，也就是zxid肯定是最大的。

由于我们的zxid都是由leader管理的，在上一节也是讲过，leader之所以能成为leader，本来就是因为它的zxid最大，此时的事务请求过来，leader的zxid本身最大的基础上再递增，这样新过来的事务的zxid肯定就是最大的。由于我们的zxid都是由leader管理的，在上一节也是讲过，leader之所以能成为leader，本来就是因为它的zxid最大，此时的事务请求过来，leader的zxid本身最大的基础上再递增，这样新过来的事务的zxid肯定就是最大的。

## zookeeper在配置中心的应用

### 1.何为配置中心

运维管理人员把配置修改之后进行提交，把配置推送到配置中心，我们分布式应用下的每个应用实例可以通过对watch事件的监听来获取配置中心的变更，在不重启服务器的情况下也能做到把应用中的一些属性从内存中换掉

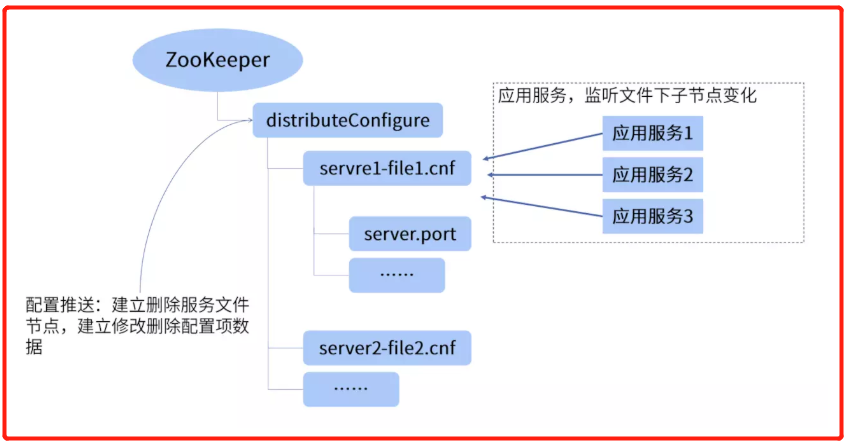


### 2.zookeeper实现配置中心

#### zookeeper配置中心的数据结构

假设我们现在拥有这么一台zookeeper服务器，我们需要创建一个配置中心的根目录distributeConfigure，注意图中的server是指我们集群中的某项服务，server1-file1.cnf是指这个服务下的配置文件1，2，3···，server.port也属于其中一个配置，从这一层开始对应zookeeper下的一个个节点，也就是说，我们把每一项配置，比如刚刚的服务端口server.port，都看作是一个znode，然后一个一个往zookeeper中存过去即可。

此时我们不关心服务是否挂掉或者怎样，也不关心节点的顺序（除非是服务功能中有特殊要求），还有就是，我们的配置一般也会有该配置的名字，但是对于顺序一般也是不要求的，所以我们就会选用持久节点来记录配置



此时应用服务要做的事情就很简单了，就是对这些个节点进行监控，只要节点存在变化，应用服务就把节点下的数据取过来即可。