

目 录

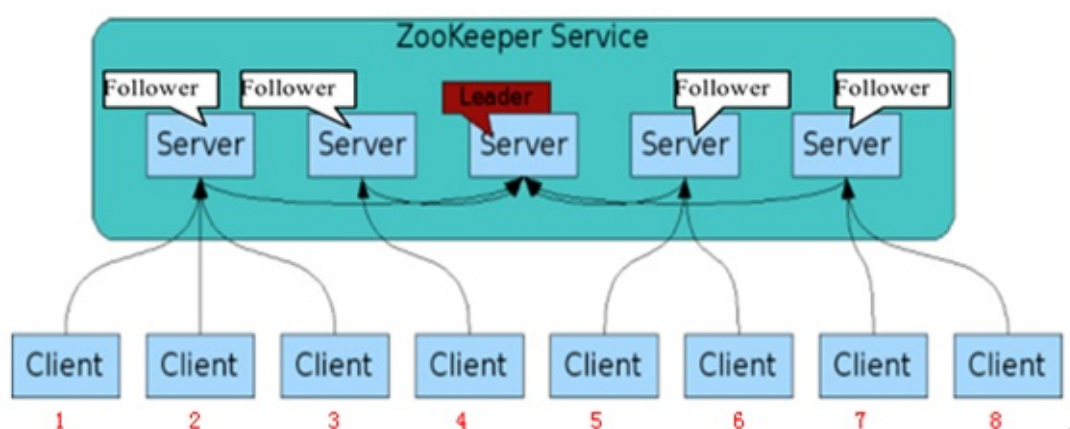
关于版块
基础知识
安装部署
zk cli操作

关于版块

关于zookeeper

zooKeeper是一种集中式服务，用于维护配置信息，命名，提供分布式同步和提供组服务！所有这些类型的服务都以分布式应用程序的某种形式使用，每次实施它们都需要做很多工作来修复不可避免的 errors 和竞争条件。这使得它们在变化的情况下变得脆弱并且难以管理。即使正确完成，这些服务的不同实现也会在部署应用程序时导致管理复杂性。ZooKeeper旨在将这些不同服务的本质提炼为一个非常简单的集中协调服务接口。

如图：



ZooKeeper提供的常见服务如下：

- 1、命名服务 - 按名称标识集群中的节点。它类似于DNS，但仅对于节点。
- 2、配置管理 - 加入节点的最近的和最新的系统配置信息。
- 3、集群管理 - 实时地在集群和节点状态中加入/离开节点。
- 4、选举算法 - 选举一个节点作为协调目的的leader。
- 5、锁定和同步服务 - 在修改数据的同时锁定数据。
- 6、高度可靠的数据注册表 - 即使在一个或几个节点关闭时也可以获得数据。

以下是使用ZooKeeper的好处：

- 1、简单的分布式协调过程
- 2、同步 - 服务器进程之间的相互排斥和协作。
- 3、有序的消息
- 4、序列化 - 根据特定规则对数据进行编码。确保应用程序运行一致。
- 5、可靠性
- 6、原子性 - 数据转移完全成功或完全失败，没有事务是部分的。

本版块维护人员

版主：子木

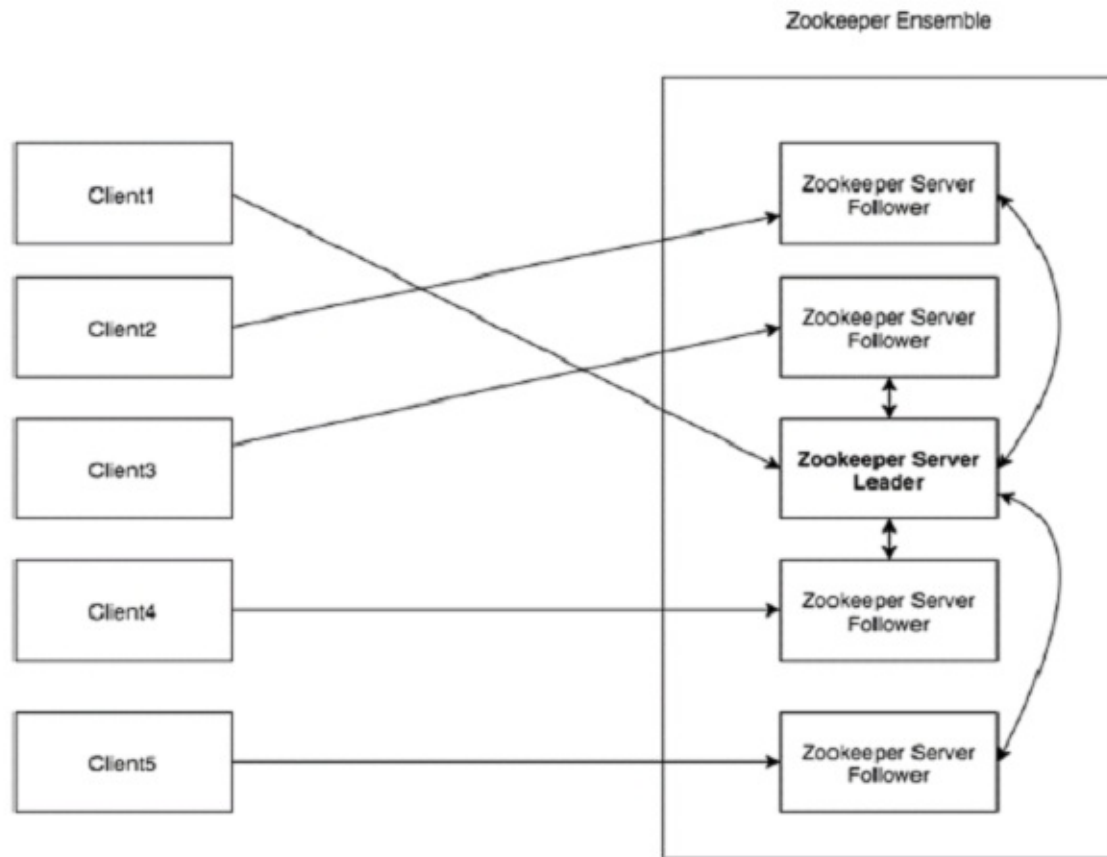
QQ：1242119478

交流Q群：526749756

基础知识

ZooKeeper的架构

ZooKeeper为“客户端-服务器”架构，如下图：



对ZooKeeper架构部分的每个组件说明如下：

Server（服务器）：服务器，我们的ZooKeeper总体中的一个节点，为客户端提供所有的服务。向客户端发送确认码以告知服务器是活跃的

Client（客户端）：客户端，我们的分布式应用集群中的一个节点，从服务器访问信息。对于特定的时间间隔，每个客户端向服务器发送消息以使服务器知道客户端是活跃的。类似地，当客户端连接时，服务器发送确认码。如果连接的服务器没有响应，客户端会自动将消息重定向到另一个服务器

Leader：服务器节点，如果任何连接的节点失败，则执行自动恢复。Leader在服务启动时被选举

Follower：跟随leader指令的服务器节点

Ensemble：ZooKeeper服务器组。形成ensemble所需的最小节点数为3

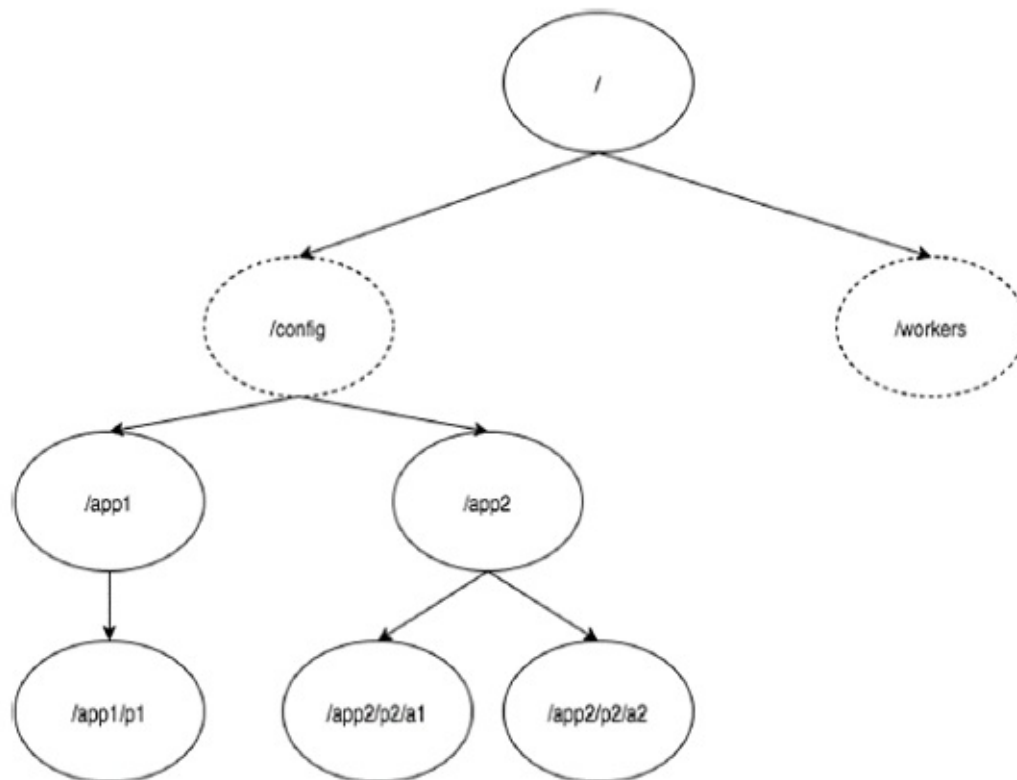
层次命名空间

下图描述了用于内存表示的ZooKeeper文件系统的树结构。ZooKeeper节点称为 **znode** 。每个znode 由一个名称标识，并用路径(/)序列分隔。

在图中，首先有一个由“/”分隔的znode。在根目录下，你有两个逻辑命名空间 **config** 和 **workers** 。

config 命名空间用于集中式配置管理，**workers** 命名空间用于命名。

在 **config** 命名空间下，每个znode最多可存储1MB的数据。这与UNIX文件系统相类似，除了父znode也可以存储数据。这种结构的主要目的是存储同步数据并描述znode的元数据。此结构称为ZooKeeper数据模型。



ZooKeeper数据模型中的每个znode都维护着一个 **stat** 结构。一个stat仅提供一个znode的元数据。它由版本号，操作控制列表(ACL)，时间戳和数据长度组成。

版本号 - 每个znode都有版本号，这意味着每当与znode相关联的数据发生变化时，其对应的版本号也会增加。当多个zookeeper客户端尝试在同一znode上执行操作时，版本号的使用就很重要。

操作控制列表(ACL) - ACL基本上是访问znode的认证机制。它管理所有znode读取和写入操作。

时间戳 - 时间戳表示创建和修改znode所经过的时间。它通常以毫秒为单位。ZooKeeper从“事务ID”(zxid)标识znode的每个更改。Zxid 是唯一的，并且为每个事务保留时间，以便你可以轻松地确定从一个请求到另一个请求所经过的时间。

数据长度 - 存储在znode中的数据总量是数据长度。你最多可以存储1MB的数据。

Znode的类型

Znode被分为持久（**persistent**）节点，顺序（**sequential**）节点和临时（**ephemeral**）节点。

持久节点 - 即使在创建该特定znode的客户端断开连接后，持久节点仍然存在。默认情况下，除非另有说明，否则所有znode都是持久的。

临时节点 - 客户端活跃时，临时节点就是有效的。当客户端与ZooKeeper集合断开连接时，临时节点会自动删除。因此，只有临时节点不允许有子节点。如果临时节点被删除，则下一个合适的节点将填充其位置。临时节点在leader选举中起着重要作用。

顺序节点 - 顺序节点可以是持久的或临时的。当一个新的znode被创建为一个顺序节点时，ZooKeeper通过将10位的序列号附加到原始名称来设置znode的路径。例如，如果将具有路径 /myapp 的znode创建为顺序节点，则ZooKeeper会将路径更改为 /myapp0000000001，并将下一个序列号设置为0000000002。如果两个顺序节点是同时创建的，那么ZooKeeper不会对每个znode使用相同的数字。顺序节点在锁定和同步中起重要作用。

Sessions（会话）

会话对于ZooKeeper的操作非常重要。会话中的请求按FIFO顺序执行。一旦客户端连接到服务器，将建立会话并向客户端分配会话ID。

客户端以特定的时间间隔发送心跳以保持会话有效。如果ZooKeeper集合在超过服务器开启时指定的期间（会话超时）都没有从客户端接收到心跳，则它会判定客户端死机。

会话超时通常以毫秒为单位。当会话由于任何原因结束时，在该会话期间创建的临时节点也会被删除。

Watches（监视）

监视是一种简单的机制，使客户端收到关于ZooKeeper集合中的更改的通知。客户端可以在读取特定znode时设置Watches。Watches会向注册的客户端发送任何znode（客户端注册表）更改的通知。

Znode更改是与znode相关的数据的修改或znode的子项中的更改。只触发一次watches。如果客户端想要再次通知，则必须通过另一个读取操作来完成。当连接会话过期时，客户端将与服务器断开连接，相关的watches也将被删除。

ZooKeeper集合中的节点

让我们分析在ZooKeeper集合中拥有不同数量的节点的效果。

如果我们有单个节点，则当该节点故障时，ZooKeeper集合将故障。它有助于“单点故障”，不建议在生产环境中使用。

如果我们有二个节点而一个节点故障，我们没有占多数，因为两个中的一个不是多数。

如果我们有三个节点而一个节点故障，那么我们有大多数，因此，这是最低要求。ZooKeeper集合在实际生产环境中必须至少有三个节点。

如果我们有四个节点而二个节点故障，它将再次故障。类似于有三个节点，额外节点不用于任何目的，因此，最好添加奇数的节点，例如3，5，7。

Zookeeper leader选举

所有节点创建具有相同路径 /app/leader_election/guid_ 的顺序、临时节点。

1、ZooKeeper集合将附加10位序列号到路径，创建的znode将是

/app/leader_election/guid_0000000001, /app/leader_election/guid_0000000002等

2、对于给定的实例，在znode中创建最小数字的节点成为leader，而所有其他节点是follower。

3、每个follower节点监视下一个具有最小数字的znode。例如，创建znode/app/leader_election/guid_0000000008的节点将监视znode/app/leader_election/guid_0000000007，创建znode/app/leader_election/guid_0000000007的节点将监视znode/app/leader_election/guid_0000000006。

4、如果leader关闭，则其相应的znode/app/leader_electionN会被删除。

5、下一个在线follower节点将通过监视器获得关于leader移除的通知。

6、下一个在线follower节点将检查是否存在其他具有最小数字的znode。如果没有，那么它将承担leader的角色。否则，它找到的创建具有最小数字的znode的节点将作为leader。

7、类似地，所有其他follower节点选举创建具有最小数字的znode的节点作为leader。

转载来自: [\[https://www.w3cschool.cn/zookeeper/zookeeper_fundamentals.html\]](https://www.w3cschool.cn/zookeeper/zookeeper_fundamentals.html)

安装部署

基础环境

服务器:

test11 192.168.37.11

test12 192.168.37.12

test13 192.168.37.13

系统版本:

centos 7.6

服务器本地安装

服务器test11下安装:

- 1、需要服务器安装好jdk环境，本例用的是zk3.4.14，需要最低JDK版本现为1.8
- 2、下载安装包: wget <http://apache.website-solution.net/zookeeper/zookeeper-3.4.14/zookeeper-3.4.14.tar.gz>
- 3、提取tar文件:

```
tar -zxvf apache-zookeeper-3.4.14.tar.gz
mv apache-zookeeper-3.4.14 /usr/local/zk-3.4.14
cd /usr/local/zk-3.4.14
mkdir data
```

- 4、创建配置文件:

```
[root@test11 apache-zookeeper-3.4.14]#vi conf/zoo.cfg
tickTime = 2000
dataDir = /usr/local/zk-3.4.14/data
clientPort = 2181
```

- 5、启动zookeeper: [root@test11 zk-3.4.14]# ./bin/zkServer.sh start
- 6、启动CLI，连接本地zk服务器上: ./bin/zkCli.sh
- 7、连接到指定的zk上: ./bin/zkCli.sh -server ip:2181
- 8、查看节点状态:

```
[root@test11 zk-3.4.14]# ./bin/zkServer.sh status
ZooKeeper JMX enabled by default
Using config: /usr/local/zk-3.4.14/bin/../conf/zoo.cfg
Mode: standalone #因为是单节点，所以显示standalone
```

- 9、停止zk: ./bin/zkServer.sh stop

安装集群

- 1、在test12,test13服务器上按照上面安装zk
- 2、test11,test12,test13这三台服务器的zoo.cfg文件配置如下：

```
[root@test12 zk-3.4.14]# cat conf/zoo.cfg
tickTime = 2000
dataDir = /usr/local/zk-3.4.14/data
clientPort = 2181
initLimit=5
syncLimit=2
server.1=192.168.37.11:2888:3888
server.2=192.168.37.12:2888:3888
server.3=192.168.37.13:2888:3888
```

- 3、给test11,test12,test13添加myid,如下：

```
[root@test11 zk-3.4.14]# echo 1 >/usr/local/zk-3.4.14/data/myid
[root@test12 zk-3.4.14]# echo 2 >/usr/local/zk-3.4.14/data/myid
[root@test13 zk-3.4.14]# echo 3 >/usr/local/zk-3.4.14/data/myid
```

- 4、启动zk服务，并查看节点状态，如下：

```
[root@test11 zk-3.4.14]# ./bin/zkServer.sh status
ZooKeeper JMX enabled by default
Using config: /usr/local/zk-3.4.14/bin/../conf/zoo.cfg
Mode: leader

[root@test12 zk-3.4.14]# ./bin/zkServer.sh status
ZooKeeper JMX enabled by default
Using config: /usr/local/zk-3.4.14/bin/../conf/zoo.cfg
Mode: follower

[root@test13 zk-3.4.14]# ./bin/zkServer.sh status
ZooKeeper JMX enabled by default
Using config: /usr/local/zk-3.4.14/bin/../conf/zoo.cfg
Mode: follower

#可以看出在上面test11是leader节点
```

注：zoo.cfg参数说明

1.tickTime: CS通信心跳数

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。

tickTime以毫秒为单位。

tickTime: 该参数用来定义心跳的间隔时间，zookeeper的客户端和服务端之间也有和web开发里类似的session的概念，而zookeeper里最小的session过期时间就是tickTime的两倍。

2.initLimit: LF初始通信时限

集群中的follower服务器(F)与leader服务器(L)之间 初始连接 时能容忍的最多心跳数（tickTime的数量）。

此配置表示，允许 follower （相对于 leader 而言的“客户端”）连接 并同步到 leader 的初始化连接时间，它以 tickTime 的倍数来表示。当超过设置倍数的 tickTime 时间，则连接失败。

3.syncLimit: LF同步通信时限

集群中的follower服务器(F)与leader服务器(L)之间 请求和应答 之间能容忍的最多心跳数（tickTime的数量）。

此配置表示，leader 与 follower 之间发送消息，请求 和 应答 时间长度。如果 follower 在设置的时间内不能与leader 进行通信，那么此 follower 将被丢弃。

docker安装

前置环境：安装好docker,并添加docker镜像加速

1、拉取镜像：`docker pull zookeeper`

2、安装docker-compose,步骤如下：

```
yum -y install epel-release
yum install python-pip -y
pip install --upgrade pip
pip install docker-compose
```

3、新建zk_3nodes.yml文件如下：

```
[root@test14 docker_compose]# cat /scripts/docker_compose/docker-compose
.yml
version: '3.5'

services:
  zk01:
    image: zookeeper
    restart: always
    hostname: zk01
    ports:
      - 2181:2181
    volumes:
      - /docker/zookeeper/data/zk01:/data
    environment:
      ZOO_MY_ID: 1
      ZOO_SERVERS: server.1=0.0.0.0:2888:3888;2181 server.2=zk02:2888:3888;2181 server.3=zk03:2888:3888;2181

  zk02:
    image: zookeeper
    restart: always
    hostname: zk02
```

```

ports:
  - 2182:2181
volumes:
  - /docker/zookeeper/data/zk02:/data
environment:
  ZOO_MY_ID: 2
  ZOO_SERVERS: server.1=zk01:2888:3888;2181 server.2=0.0.0.0:2888:3888;2181 server.3=zk03:2888:3888;2181

zk03:
  image: zookeeper
  restart: always
  hostname: zk03
  ports:
    - 2183:2181
  volumes:
    - /docker/zookeeper/data/zk03:/data
  environment:
    ZOO_MY_ID: 3
    ZOO_SERVERS: server.1=zk01:2888:3888;2181 server.2=zk02:2888:3888;2181 server.3=0.0.0.0:2888:3888;2181

```

4、创建集群启动脚本

```

[root@test14 docker_compose]# cat /scripts/zk_3nodes_start.sh
#!/bin/bash
DataPath=/docker/zookeeper/data

for i in {1..3} ; do

[ -d $DataPath/zk0${i} ] || mkdir $DataPath/zk0${i} -p

done

cd /scripts/docker_compose

COMPOSE_PROJECT_NAME=zk_3nodes docker-compose up -d

```

zk cli操作

管理ZooKeeper存储

创建Znodes

- 1、连接到ZooKeeper: `bin/zkCli.sh -server 127.0.0.1:2181`
- 2、用给定的路径创建一个znode。flag参数指定创建的znode是临时的，持久的还是顺序的。默认情况下，所有znode都是持久的
 ——当会话过期或客户端断开连接时，临时节点（flag: -e）将被自动删除
 ——顺序节点保证znode路径将是唯一的，ZooKeeper集合将向znode路径填充10位序列号。例如，znode路径 /myapp 将转换为/myapp0000000001，下一个序列号将为/myapp0000000002。
- 3、创建一个新的持久的znode并将字符串“first-data”与节点相关联

```
[zk: localhost:2181(CONNECTED) 0] create /FirstZnode "first-data"
Created /FirstZnode
[zk: localhost:2181(CONNECTED) 1] get /FirstZnode
first-data
cZxid = 0x1000000002
ctime = Wed Jul 03 18:57:13 CST 2019
mZxid = 0x1000000002
mtime = Wed Jul 03 18:57:13 CST 2019
pZxid = 0x1000000002
cversion = 0
dataVersion = 0
aclVersion = 0
ephemeralOwner = 0x0
dataLength = 10
numChildren = 0
```

- 4、创建一个临时的znode并将字符串“temp-data”与节点相关联

```
[zk: localhost:2181(CONNECTED) 7] create -e /SecondZnode "temp-data"
Created /SecondZnode
```

- 5、创建一个顺序的znode并将字符串“order-data”与节点相关联

```
[zk: localhost:2181(CONNECTED) 12] create -s /ThirdZnode "order-data"
Created /ThirdZnode0000000003
```

管理znodes数据

- 6、发出另一个ls /命令以查看目录的外观

```
[zk: localhost:2181(CONNECTED) 6] ls /
[zookeeper, FirstZnode]
```

7、通过运行get命令验证数据是否与znode相关联

```
[zk: localhost:2181(CONNECTED) 1] get /FirstZnode
first-data
cZxid = 0x1000000002
ctime = Wed Jul 03 18:57:13 CST 2019
mZxid = 0x1000000002
mtime = Wed Jul 03 18:57:13 CST 2019
pZxid = 0x1000000002
cversion = 0
dataVersion = 0
aclVersion = 0
ephemeralOwner = 0x0
dataLength = 10
numChildren = 0
```

注意：如果是顺序的znode的需要get全路径，例如：get /ThirdZnode0000000003

8、通过发出set命令来更改与zk_test关联的数据

```
[zk: localhost:2181(CONNECTED) 3] set /FirstZnode "lasting-data"
cZxid = 0x1000000002
ctime = Wed Jul 03 18:57:13 CST 2019
mZxid = 0x1000000004
mtime = Wed Jul 03 18:58:54 CST 2019
pZxid = 0x1000000002
cversion = 0
dataVersion = 1
aclVersion = 0
ephemeralOwner = 0x0
dataLength = 12
numChildren = 0
```

创建子项/子节点

创建子节点类似于创建新的znode。唯一的区别是，子znode的路径也将具有父路径。

例如在FirstZnode节点下再创建Child1节点：

```
[zk: localhost:2181(CONNECTED) 21] create /FirstZnode/child1 "child1-data"
Created /FirstZnode/child1
[zk: localhost:2181(CONNECTED) 23] ls /FirstZnode
[child1]
```

检查状态

状态描述指定的znode的元数据。它包含时间戳，版本号，ACL，数据长度和子znode等细项

```
[zk: localhost:2181(CONNECTED) 24] stat /FirstZnode
cZxid = 0x1000000002
ctime = Wed Jul 03 18:57:13 CST 2019
mZxid = 0x1000000004
mtime = Wed Jul 03 18:58:54 CST 2019
pZxid = 0x100000000c
cversion = 1
dataVersion = 1
aclVersion = 0
ephemeralOwner = 0x0
dataLength = 12
numChildren = 1
```

删除节点

删除没有子节点的节点: `delete /SecondZnode`

删除有子节点的节点需要用: `rmr /FirstZnode`