# 一：简介

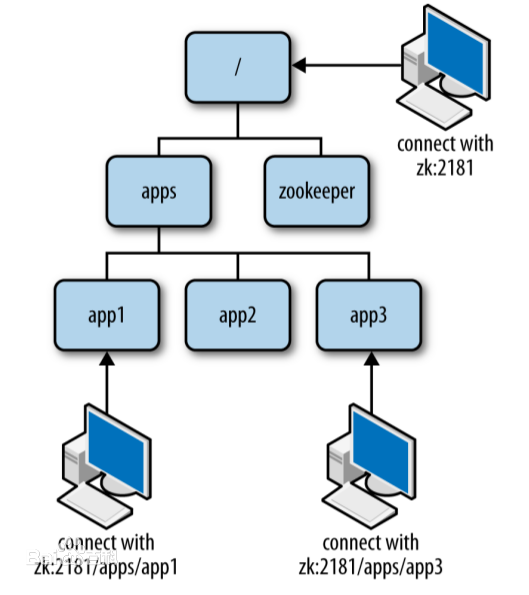
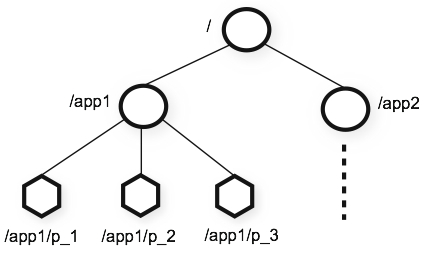
分布式应用程序可以基于Zookeeper实现例如配置管理、数据发布/订阅、负载均衡、命名服务、协调通知、集群管理、Master选举、分布式锁、分布式队列等功能。

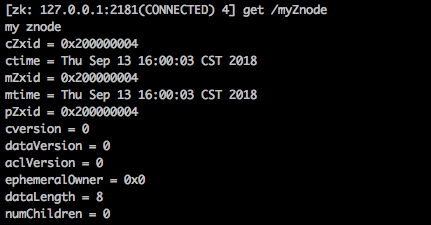
<https://zookeeper.apache.org/>

###Zookeeper中重要的概念：

## 数据节点：

Zookeeper存储数据是以树形结构存储的，类似linux的目录，最上层是“/”也就是根节点，可以在根节点上创建子节点，在子节点上再创建子节点。在Zookeeper中每个节点被称为一个Znode,每个Znode包含：节点名称、节点值、值的长度、节点创建时间、节点修改时间、版本号、子节点数量等属性，可以创建节点，修改节点，删除节点等操作。





## Watcher(事件监听器)

事件监听是Zookeeper中非常重要的一个特性，用来监听节点的操作，当对节点做了操作(例如创建节点、修改节点、删除节点等)就会触发节点上的监听，当监听到节点发生了变化就可以做一些相应的处理操作。该

机制是ZooKeeper实现分布式协调服务的重要特性。

## 集群角色

Leader : 主节点，就是我们平时所说的master节点，用于提供读写服务

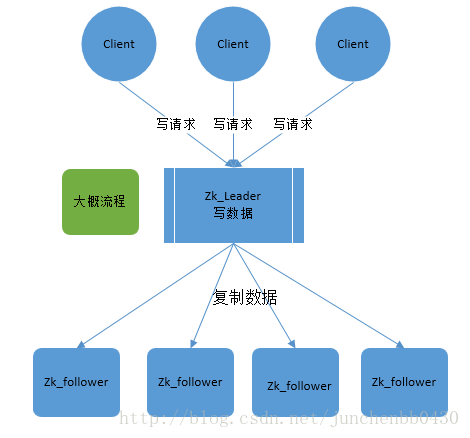
Follower: 从节点，就是我们平时所说的slaver节点，提供读服务、leader选举等

Observer: 从节点，一种特殊的Follower, 提供读服务, 不参与leader的选举

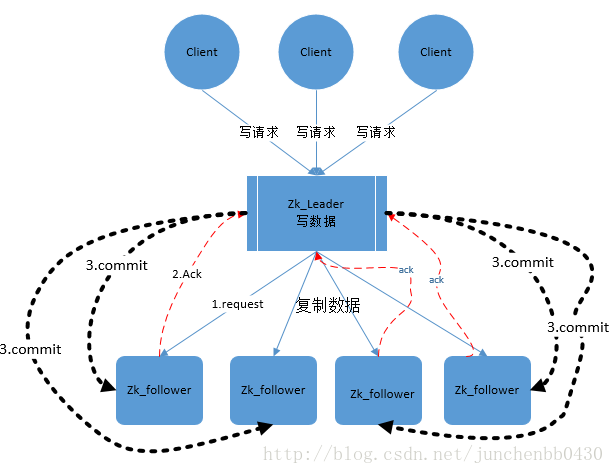
## ZAB协议

ZAB协议是专门为zookeeper实现分布式协调功能而设计。zookeeper主要是根据ZAB协议是实现分布式系统数据一致性。

zookeeper根据ZAB协议建立了主备模型完成zookeeper集群中数据的同步。这里所说的主备系统架构模型是指，在zookeeper集群中，只有一台leader负责处理外部客户端的事物请求(或写操作)，然后leader服务器将客户端的写操作数据同步到所有的follower节点中。



ZAB的协议核心是在整个zookeeper集群中只有一个节点即Leader将客户端的写操作转化为事物(或提议proposal)。Leader节点再数据写完之后，将向所有的follower节点发送数据广播请求(或数据复制)，等待所有的follower节点反馈。在ZAB协议中，只要超过半数follower节点反馈OK，Leader节点就会向所有的follower服务器发送commit消息。即将leader节点上的数据同步到follower节点之上。



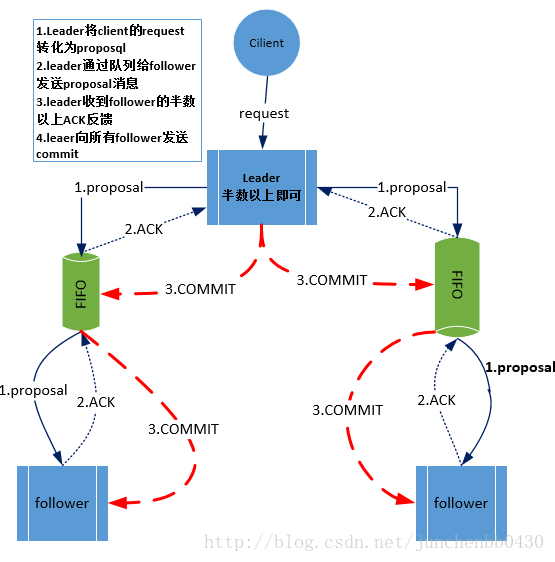
ZAB协议中主要有两种模式，第一是消息广播模式；第二是崩溃恢复模式

### 消息广播模式

1. 在zookeeper集群中数据副本的传递策略就是采用消息广播模式。zookeeper中数据副本的同步方式与二阶段提交相似但是却又不同。二阶段提交的要求协调者必须等到所有的参与者全部反馈ACK确认消息后，再发送commit消息。要求所有的参与者要么全部成功要么全部失败。二阶段提交会产生严重阻塞问题。

2. ZAB协议中Leader等待follower的ACK反馈是指”只要半数以上的follower成功反馈即可，不需要收到全部follower反馈”

3. 图中展示了消息广播的具体流程图



**zookeeper中消息广播的具体步骤如下**：

4.1. 客户端发起一个写操作请求

4.2. Leader服务器将客户端的request请求转化为事物proposql提案，同时为每个proposal分配一个全局唯一的ID，即ZXID。

4.3. leader服务器与每个follower之间都有一个队列，leader将消息发送到该队列

4.4. follower机器从队列中取出消息处理完(写入本地事物日志中)毕后，向leader服务器发送ACK确认。

4.5. leader服务器收到半数以上的follower的ACK后，即认为可以发送commit

4.6. leader向所有的follower服务器发送commit消息。

**zookeeper采用ZAB协议的核心就是只要有一台服务器提交了proposal，就要确保所有的服务器最终都能正确提交proposal。这也是CAP/BASE最终实现一致性的一个体现**。

**leader服务器与每个follower之间都有一个单独的队列进行收发消息，使用队列消息可以做到异步解耦。leader和follower之间只要往队列中发送了消息即可。如果使用同步方式容易引起阻塞。性能上要下降很多**。

### 崩溃恢复

1 zookeeper集群中为保证任何所有进程能够有序的顺序执行，只能是leader服务器接受写请求，即使是follower服务器接受到客户端的请求，也会转发到leader服务器进行处理。

2 如果leader服务器发生崩溃，则zab协议要求zookeeper集群进行崩溃恢复和leader服务器选举。

3 ZAB协议崩溃恢复要求满足如下2个要求：

3.1. 确保已经被leader提交的proposal必须最终被所有的follower服务器提交。

3.2. **确保丢弃已经被leader出的但是没有被提交的proposal**。

根据上述要求，新选举出来的leader不能包含未提交的proposal，即新选举的leader必须都是已经提交了的proposal的follower服务器节点。同时，新选举的leader节点中含有最高的ZXID。这样做的好处就是可以避免了leader服务器检查proposal的提交和丢弃工作。

leader服务器发生崩溃时分为如下场景：

5.1. leader在提出proposal时未提交之前崩溃，则经过崩溃恢复之后，新选举的leader一定不能是刚才的leader。因为这个leader存在未提交的proposal。

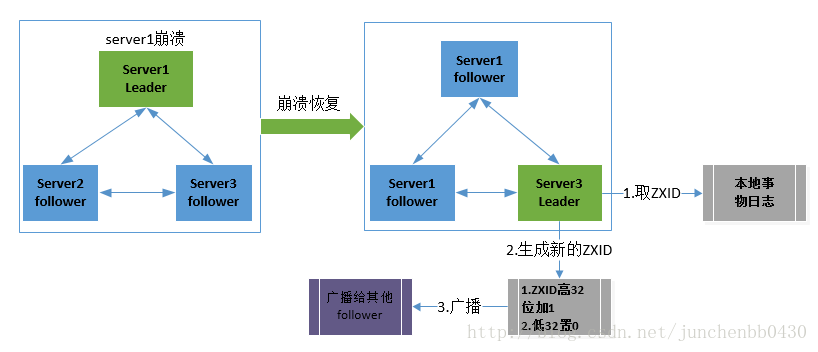
5.2 leader在发送commit消息之后，崩溃。即消息已经发送到队列中。经过崩溃恢复之后，参与选举的follower服务器(刚才崩溃的leader有可能已经恢复运行，也属于follower节点范畴)中有的节点已经是消费了队列中所有的commit消息。即该follower节点将会被选举为最新的leader。剩下动作就是数据同步过程。

#### 数据同步

1 在zookeeper集群中新的leader选举成功之后，leader会将自身的提交的最大proposal的事物ZXID发送给其他的follower节点。follower节点会根据leader的消息进行回退或者是数据同步操作。最终目的要保证集群中所有节点的数据副本保持一致。

数据同步完之后，zookeeper集群如何保证新选举的leader分配的ZXID是全局唯一呢？这个就要从ZXID的设计谈起。

2 2.1 ZXID是一个长度64位的数字，其中低32位是按照数字递增，即每次客户端发起一个proposal,低32位的数字简单加1。高32位是leader周期的epoch编号，至于这个编号如何产生(我也没有搞明白)，每当选举出一个新的leader时，新的leader就从本地事物日志中取出ZXID,然后解析出高32位的epoch编号，进行加1，再将低32位的全部设置为0。这样就保证了每次新选举的leader后，保证了ZXID的唯一性而且是保证递增的。



### ZAB协议原理

ZAB协议要求每个leader都要经历三个阶段，即发现，同步，广播。

**发现**：即要求zookeeper集群必须选择出一个leader进程，同时leader会维护一个follower可用列表。将来客户端可以这follower中的节点进行通信。

**同步**：leader要负责将本身的数据与follower完成同步，做到多副本存储。这样也是体现了CAP中高可用和分区容错。follower将队列中未处理完的请求消费完成后，写入本地事物日志中。

**广播**：leader可以接受客户端新的proposal请求，将新的proposal请求广播给所有的follower。

### Zookeeper设计目标

1 zookeeper作为当今最流行的分布式系统应用协调框架，采用zab协议的最大目标就是建立一个高可用可扩展的分布式数据主备系统。即在任何时刻只要leader发生宕机，都能保证分布式系统数据的可靠性和最终一致性。

2 深刻理解ZAB协议，才能更好的理解zookeeper对于分布式系统建设的重要性。以及为什么采用zookeeper就能保证分布式系统中数据最终一致性，服务的高可用性。

## Paxos 算法

[https://baike.baidu.com/item/Paxos 算法/10688635?fr=aladdin](https://baike.baidu.com/item/Paxos%20%E7%AE%97%E6%B3%95/10688635?fr=aladdin)

简介相关文章：<https://www.jianshu.com/p/8bf3b7ce3eaa>

# 二：单机搭建zookeeper集群

所谓单机搭建zookeeper集群其实就是在一台机器上启动多个zookeeper，在启动每个zookeeper时分别使用不同的配置文件zoo.cfg来启动,每个配置文件使用不同的配置参数(clientPort端口号、dataDir数据目录、dataLogDir数据日志目录)在同一台机器上启动多次。

## 1. 配置多个zoo.cfg配置文件

zoo.cfg默认在/usr/local/etc/zookeeper路径下(不同的操作系统路径不一样，这里用的是mac的路径)，在这个路径下创建zoo1.cfg、zoo2.cfg、zoo3.cfg三个配置文件。

zoo1.cfg

# The number of milliseconds of each tick

tickTime=2000

# The number of ticks that the initial

# synchronization phase can take

initLimit=10

# The number of ticks that can pass between

# sending a request and getting an acknowledgement

syncLimit=5

# the directory where the snapshot is stored.

# do not use /tmp for storage, /tmp here is just

# example sakes.

# 数据目录

dataDir=/usr/local/var/run/zookeeper/zk1/data

dataLogDir=/usr/local/var/run/zookeeper/zk1/logs

# the port at which the clients will connect

# 端口号

clientPort=2181

# the maximum number of client connections.

# increase this if you need to handle more clients

#maxClientCnxns=60

#

# Be sure to read the maintenance section of the

# administrator guide before turning on autopurge.

#

# http://zookeeper.apache.org/doc/current/zookeeperAdmin.html#sc\_maintenance

#

# The number of snapshots to retain in dataDir

#autopurge.snapRetainCount=3

# Purge task interval in hours

# Set to "0" to disable auto purge feature

#autopurge.purgeInterval=1

server.1=localhost:2287:3387

server.2=localhost:2288:3388

server.3=localhost:2289:3389

zoo2.cfg

# The number of milliseconds of each tick

tickTime=2000

# The number of ticks that the initial

# synchronization phase can take

initLimit=10

# The number of ticks that can pass between

# sending a request and getting an acknowledgement

syncLimit=5

# the directory where the snapshot is stored.

# do not use /tmp for storage, /tmp here is just

# example sakes.

dataDir=/usr/local/var/run/zookeeper/zk2/data

dataLogDir=/usr/local/var/run/zookeeper/zk2/logs

# the port at which the clients will connect

clientPort=2182

# the maximum number of client connections.

# increase this if you need to handle more clients

#maxClientCnxns=60

#

# Be sure to read the maintenance section of the

# administrator guide before turning on autopurge.

#

# http://zookeeper.apache.org/doc/current/zookeeperAdmin.html#sc\_maintenance

#

# The number of snapshots to retain in dataDir

#autopurge.snapRetainCount=3

# Purge task interval in hours

# Set to "0" to disable auto purge feature

#autopurge.purgeInterval=1

server.1=localhost:2287:3387

server.2=localhost:2288:3388

server.3=localhost:2289:3389

zoo3.cfg

# The number of milliseconds of each tick

tickTime=2000

# The number of ticks that the initial

# synchronization phase can take

initLimit=10

# The number of ticks that can pass between

# sending a request and getting an acknowledgement

syncLimit=5

# the directory where the snapshot is stored.

# do not use /tmp for storage, /tmp here is just

# example sakes.

dataDir=/usr/local/var/run/zookeeper/zk3/data

dataLogDir=/usr/local/var/run/zookeeper/zk3/logs

# the port at which the clients will connect

clientPort=2183

# the maximum number of client connections.

# increase this if you need to handle more clients

#maxClientCnxns=60

#

# Be sure to read the maintenance section of the

# administrator guide before turning on autopurge.

#

# http://zookeeper.apache.org/doc/current/zookeeperAdmin.html#sc\_maintenance

#

# The number of snapshots to retain in dataDir

#autopurge.snapRetainCount=3

# Purge task interval in hours

# Set to "0" to disable auto purge feature

#autopurge.purgeInterval=1

server.1=localhost:2287:3387

server.2=localhost:2288:3388

server.3=localhost:2289:3389

tickTime: 基本事件单元，以毫秒为单位。这个时间是作为zookeeper服务器之间或客户端与服务器间维持心跳的时间。也就是每隔tickTime时间就会发送一个心跳

dataDir: 存储内存中数据库快照的位置，就是zookeeper保存数据的目录，默认情况下，zookeeper将数据的日志问也保存在这个目录里

clientPort: 客户端连接zookeeper服务器的端口，默认是2181，zookeeper会监听这个端口，接收客户端的访问请求

initLimit: 这个配置项是用来配置zookeeper接收客户端初始化连接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(tickTime)长度后，zookeeper服务器还没有接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: 这个配置项标识Leader和Follower之间发送消息，请求和应答的长度，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

server.myid=IP:Port1:Port2, myid是服务器的编号，一个正整数，一般是0、1、2、3等待，port1表示的是服务器与集群中的Leader服务器交换信息的端口，一般用2288，Port2表示的是万一集群中的Leader服务器宕机了，需要一个端口来重新进行宣讲，选出一个新的Leader，一般用3388

## 2. 创建data目录和logs目录

创建/usr/local/var/run/zookeeper/zk1/data/myid文件

1

创建/usr/local/var/run/zookeeper/zk2/data/myid文件

2

创建/usr/local/var/run/zookeeper/zk3/data/myid文件

3

在zk1、zk2、zk3下创建一个logs目录

# 三：分别启动3个zookeeper

# 启动3个zookeeper服务

zkServer start /usr/local/etc/zookeeper/zoo1.cfg

zkServer start /usr/local/etc/zookeeper/zoo2.cfg

zkServer start /usr/local/etc/zookeeper/zoo3.cfg

# 查看每个zookeeper对应的角色

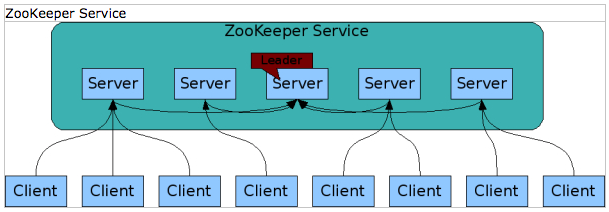
zkServer status /usr/local/etc/zookeeper/zoo1.cfg

zkServer status /usr/local/etc/zookeeper/zoo2.cfg

zkServer status /usr/local/etc/zookeeper/zoo3.cfg

# 停止zookeeper服务

zkServer stop /usr/local/etc/zookeeper/zoo1.cfg





# 四：ZooKeeper命令行

# 连接服务器 zkCli -server IP:PORT

zkCli -server 127.0.0.1:2181

# 帮助命令

help

# 获取根节点列表，默认会有一个zookeeper节点，节点列表就像linux目录一样，根节点为/

ls /

# 创建znode并关联值, create /节点名称 “值”

create /myZnode "my znode"

# 获取znode

get /myZnode

# 修改znode

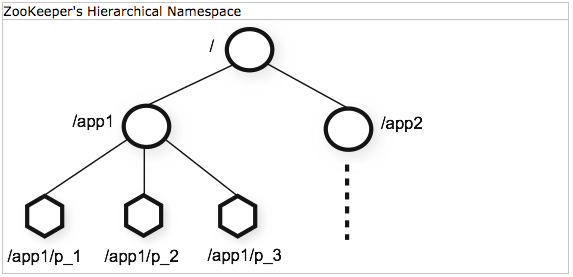
set /myZnode "my znode value string"

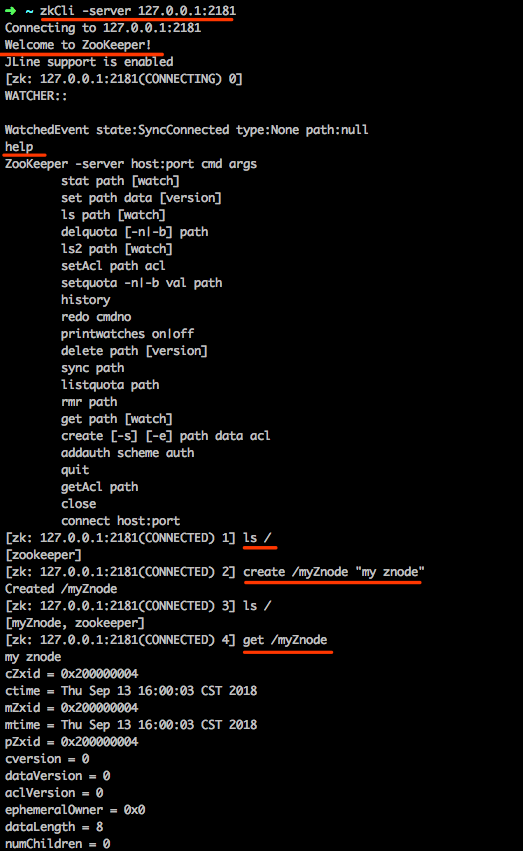
# 删除单个znode

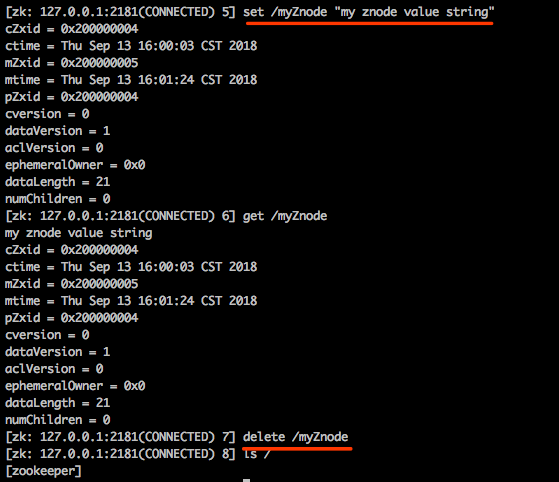
delete /myZnode

# 递归删除(删除myZnode的子节点和myZnode节点)

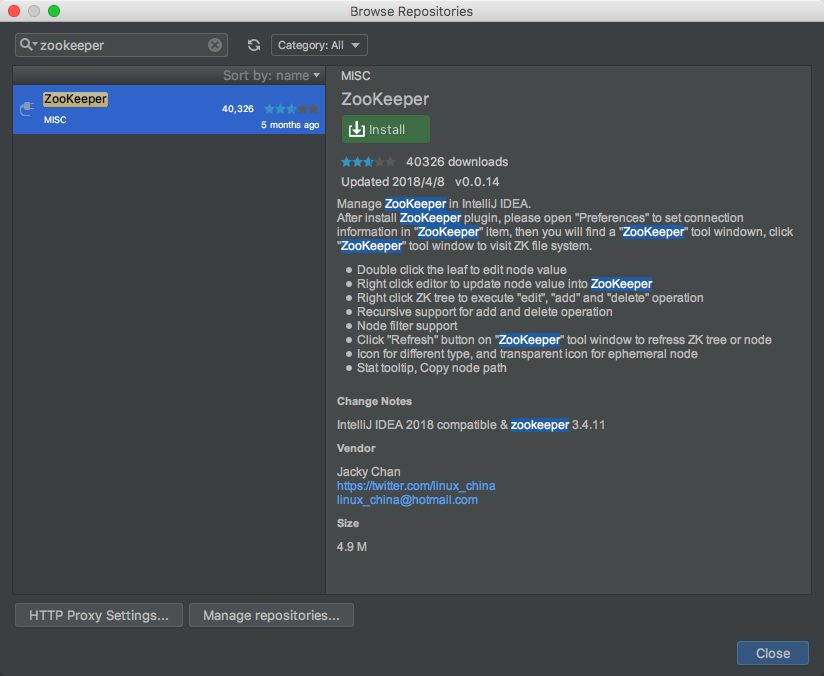
rmr /myZnode

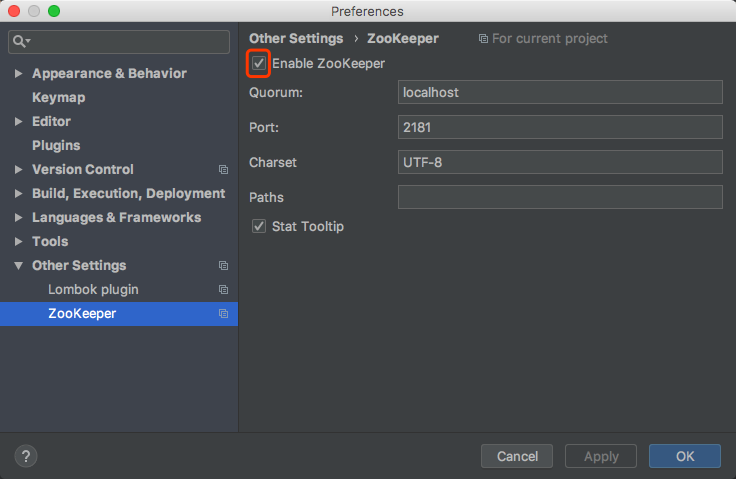


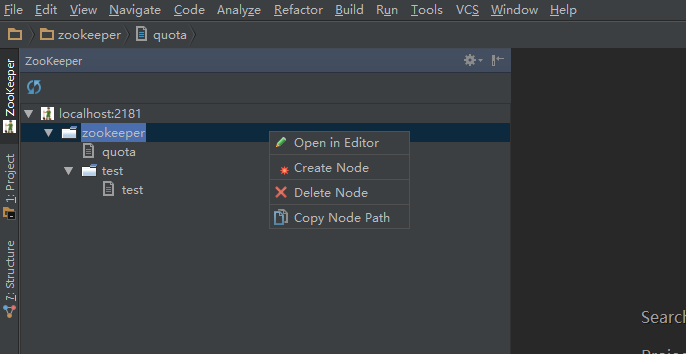




IDEA Zookeeper插件







# 五：命令

ZooKeeper 支持某些特定的四字命令字母与其的交互。它们大多是查询命令，用来获取 ZooKeeper 服务的当前状态及相关信息。用户在可以通过 telnet 或 nc 向 ZooKeeper 提交相应的命令。

|  |  |
| --- | --- |
| **ZooKeeper 四字命令** | **功能描述** |
| conf | 输出相关服务配置的详细信息。 |
| cons | 列出所有连接到服务器的客户端的完全的连接 / 会话的详细信息。包括“接受 / 发送”的包数量、会话 id 、操作延迟、最后的操作执行等等信息。 |
| dump | 列出未经处理的会话和临时节点。 |
| envi | 输出关于服务环境的详细信息（区别于 conf 命令）。 |
| reqs | 列出未经处理的请求 |
| ruok | 测试服务是否处于正确状态。如果确实如此，那么服务返回“imok ”，否则不做任何相应。 |
| stat | 输出关于性能和连接的客户端的列表。 |
| wchs | 列出服务器 watch 的详细信息。 |
| wchc | 通过 session 列出服务器 watch 的详细信息，它的输出是一个与watch 相关的会话的列表。 |
| wchp | 通过路径列出服务器 watch 的详细信息。它输出一个与 session相关的路径。 |

