# Zookeeper操作手册

# 安装

## 下载

<http://apache.website-solution.net/zookeeper/stable/>

## 安装

--bin

--conf

--contrib

--dist-maven

--docs

--lib

--recipes

--src

解压即可

# 配置

所有的配置文件都在conf目录下

## zoo.cfg

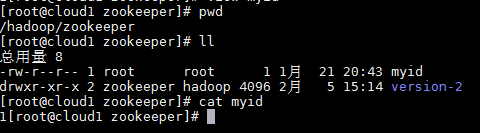
|  |
| --- |
| clientPort=2181  initLimit=10  autopurge.purgeInterval=24  syncLimit=5  tickTime=2000  dataDir=/hadoop/zookeeper  autopurge.snapRetainCount=30  server.1=cloud1.cd121.cc:2888:3888  server.2=cloud2.cd121.cc:2888:3888  server.3=cloud3.cd121.cc:2888:3888 |

这里配置了三台zk，分别是server1，server2，server3

注意这里的dataDir

在dataDir目录下输出myid文件

例如cloud1.cd121.cc这台机器myid就是1，其他依次



## JVM配置

在bin/目录下放zookeeper-env.sh

|  |
| --- |
| export JAVA\_HOME=/usr/java/jdk1.8.0\_101  **export ZOOKEEPER\_HOME=/usr/hdp/current/zookeeper-server**  **export ZOO\_LOG\_DIR=/var/log/zookeeper**  **export ZOO\_LOG4J\_PROP="INFO,ROLLINGFILE"**  export ZOOPIDFILE=/var/run/zookeeper/zookeeper\_server.pid  **export SERVER\_JVMFLAGS=-Xmx1024m**  export JAVA=$JAVA\_HOME/bin/java  export CLASSPATH=$CLASSPATH:/usr/share/zookeeper/\* |

暂时只关心红色部分，请按实际情况配置修改

## 日志配置

默认的log4j配置文件，很多配置不是很合理，建议改用下面的

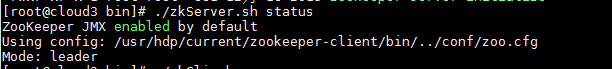
|  |
| --- |
| # Define some default values that can be overridden by system properties  zookeeper.root.logger=INFO,ROLLINGFILE  zookeeper.console.threshold=INFO  zookeeper.log.dir=.  zookeeper.log.file=zookeeper-1.log  zookeeper.log.threshold=DEBUG  zookeeper.tracelog.dir=.  zookeeper.tracelog.file=zookeeper\_trace.log  #  # ZooKeeper Logging Configuration  #  # Format is "<default threshold> (, <appender>)+  # DEFAULT: console appender only  log4j.rootLogger=${zookeeper.root.logger}  # Example with rolling log file  #log4j.rootLogger=DEBUG, CONSOLE, ROLLINGFILE  # Example with rolling log file and tracing  #log4j.rootLogger=TRACE, CONSOLE, ROLLINGFILE, TRACEFILE  #  # Log INFO level and above messages to the console  #  log4j.appender.CONSOLE=org.apache.log4j.ConsoleAppender  log4j.appender.CONSOLE.Threshold=${zookeeper.console.threshold}  log4j.appender.CONSOLE.layout=org.apache.log4j.PatternLayout  log4j.appender.CONSOLE.layout.ConversionPattern=%d{ISO8601} [myid:%X{myid}] - %-5p [%t:%C{1}@%L] - %m%n  #  # Add ROLLINGFILE to rootLogger to get log file output  # Log DEBUG level and above messages to a log file  log4j.appender.ROLLINGFILE=org.apache.log4j.DailyRollingFileAppender  log4j.appender.ROLLINGFILE.Threshold=${zookeeper.log.threshold}  log4j.appender.ROLLINGFILE.File=${zookeeper.log.dir}/${zookeeper.log.file}  # Max log file size of 10MB  #log4j.appender.ROLLINGFILE.MaxFileSize=10MB  # uncomment the next line to limit number of backup files  #log4j.appender.ROLLINGFILE.MaxBackupIndex=10  log4j.appender.ROLLINGFILE.layout=org.apache.log4j.PatternLayout  log4j.appender.ROLLINGFILE.layout.ConversionPattern=%d{ISO8601} [myid:%X{myid}] - %-5p [%t:%C{1}@%L] - %m%n  #  # Add TRACEFILE to rootLogger to get log file output  # Log DEBUG level and above messages to a log file  log4j.appender.TRACEFILE=org.apache.log4j.FileAppender  log4j.appender.TRACEFILE.Threshold=TRACE  log4j.appender.TRACEFILE.File=${zookeeper.tracelog.dir}/${zookeeper.tracelog.file}  log4j.appender.TRACEFILE.layout=org.apache.log4j.PatternLayout  ### Notice we are including log4j's NDC here (%x)  log4j.appender.TRACEFILE.layout.ConversionPattern=%d{ISO8601} [myid:%X{myid}] - %-5p [%t:%C{1}@%L][%x] - %m%n |

因为Zookeeper要频繁的写**txlog**(Zookeeper写的一种顺序日志)以及定期dump内存snapshot到磁盘，这样磁盘占用就越来越大，所以Zookeeper提供了清理这些文件的机制，但是这种机制并不太合理，它只能设置间隔多久清理，而不能设置具体的时间段。那么就有可能碰到高峰期间清理，所以建议将其关闭:autopurge.purgeInterval=0。然后使用crontab等机制，在业务低谷的时候清理。

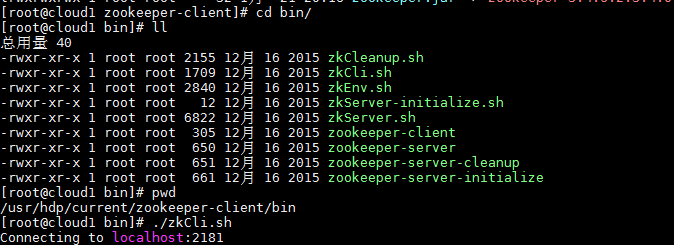
三种IO: txlog(每个写操作，包括新Session都会记录一条log)，Snapshot以及运行的应用日志

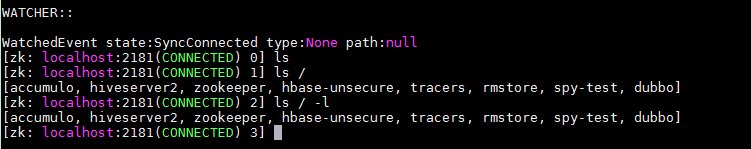
# 常用命令

|  |
| --- |
| 启动：./zkServer.sh start  停止：./zkServer.sh stop  当前状态：./zkServer.sh status |



常用shell





1. 显示根目录下、文件： ls / 使用 ls 命令来查看当前 ZooKeeper 中所包含的内容

2. 显示根目录下、文件： ls2 / 查看当前节点数据并能看到更新次数等数据

3. 创建文件，并设置初始内容： create /zk “test” 创建一个新的 znode节点“ zk ”以及与它关联的字符串

4. 获取文件内容： get /zk 确认 znode 是否包含我们所创建的字符串

5. 修改文件内容： set /zk “zkbak” 对 zk 所关联的字符串进行设置

6. 删除文件： delete /zk 将刚才创建的 znode 删除

7. 删除包含子文件夹：rmr /zk 将创建的zk目录已经它下面的子目录都删除

8. 退出客户端： quit

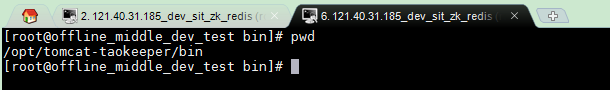
9. 帮助命令： help

# 管理界面UI

## taokeeper

<https://github.com/alibaba/taokeeper>

构建时注意依赖jar，common—tookit，json，zkclient，或者直接下载已构建好的包



好像没什么用

## zkui

<https://github.com/DeemOpen/zkui>

安装

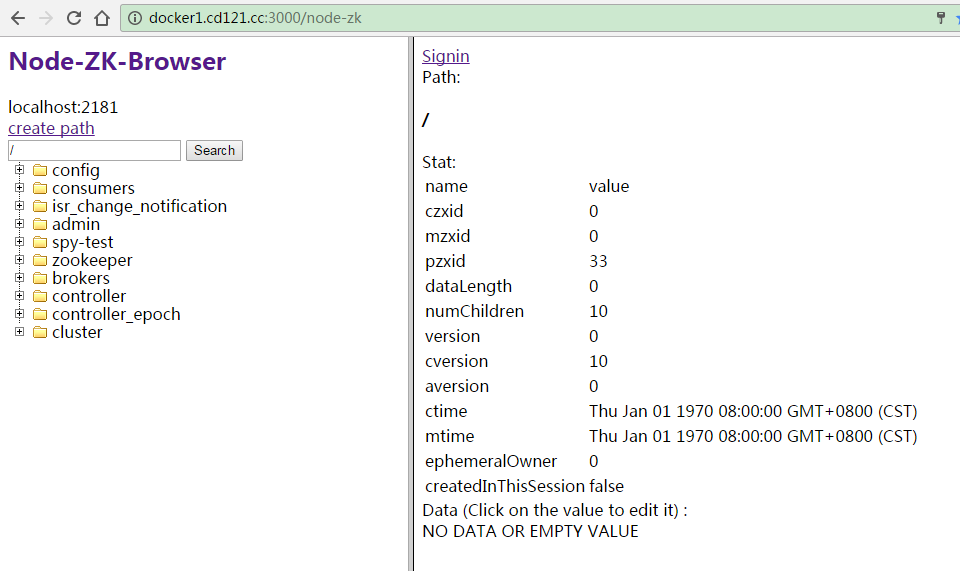
http://120.55.85.150:9090/

admin/manager 管理员权限

appconfig/appconfig 只读权限

## Node-ZK-Browser（推荐）

http://docker1.cd121.cc:3000/node-zk



## zkdash

也是一个不错的可视化UI

https://github.com/ireaderlab/zkdash

# 特性

**数据发布订阅**：对统一配置信息等数据可以通过在Zookeeper创建一个数据节点并让客户端进行监听，主要利用了Zookeeper的Watcher监听特性；

**负载均衡**：创建一个节点，负载应用把自己的服务地址写到此节点下，如果此应用挂掉，则此子节点消失

**命名服务**：利用Zookeeper创建顺序无重复子节点的特性；

**分布式协调/通知**：不同客户端都对Zookeeper上的同一个数据节点进行watcher注册，监听数据节点的变化，当发生变化所有订阅的客户端接收到通知并进行处理；

**集群管理**：利用了watcher监听与临时节点在会话失效自动清除的特性。同时，各服务器可以讲运行状态信息写入到临时节点中进而有助于Leader收集负载信息；

**Master选举**：所有客户端创建同一个path的数据节点，只有一个能成功，即为Master；

**分布式锁**：创建临时节点，谁成功即获得锁。另外，根据创建时不同的类型-序号，根据一定的规则可以模拟出共享锁、读写锁等；

**分布式队列**：每个客户端在指定节点下创建临时节点，然后获取该指定节点下的所有子节点并判断自己是否是序号最小的节点，如果是则可以进行处理，如果不是则进入等待并监听比自己序号小的最后一个节点，待接到watcher通知后，重复检查。

# 开发注意事项

* 节点数据不宜太大（<1M）

# 性能评估

# 参考资料