**ZooKeeper 入门指南**



目录

[1 Getting Started: ZooKeeper协调分布式应用程序 2](#_Toc501906325)

[1.1 前置必备 2](#_Toc501906326)

[1.2 下载 2](#_Toc501906327)

[1.3 单节点操作 2](#_Toc501906328)

[tickTime 2](#_Toc501906329)

[dataDir 2](#_Toc501906330)

[clientPort 3](#_Toc501906331)

[1.4 ZooKeeper存储库 4](#_Toc501906332)

[1.5 连接 ZooKeeper 4](#_Toc501906333)

[1.6 用 ZooKeeper编程 7](#_Toc501906334)

[1.7 运行赋值的ZooKeeper 8](#_Toc501906335)

[1.8 其他优化 9](#_Toc501906336)

# Getting Started: ZooKeeper协调分布式应用程序

这份文档包含的信息，让你迅速开始使用ZooKeeper。它主要针对的是开发人员希望它，并包含简单的单节点的安装说明，一些命令，以验证它正在运行，和一个简单的编程示例。最后为了方便起见，有几个部分设计更复杂的安装，例如运行赋值的部署，以及优化事务日志。但是，对于商业部署的完整说明，请参考: [ZooKeeper 管理指南](https://zookeeper.apache.org/doc/current/zookeeperAdmin.html)。

## 前置必备

请参考管理指南中的[系统要求](https://zookeeper.apache.org/doc/current/zookeeperAdmin.html#sc_systemReq)。

## 下载

要获取ZooKeeper，请从一个Apache下载镜像中[下载](https://zookeeper.apache.org/doc/current/zookeeperAdmin.html#sc_systemReq)最近的稳定版本。

## 单节点操作

在单节点模式下设置ZooKeeper服务器非常简单。服务器包含在单个JAR文件中，因此安装包括创建配置。一旦你下载了一个稳定的ZooKeeper，解压和cd到根目录。

tickTime=2000 dataDir=/var/lib/zookeeper clientPort=2181

要启动ZooKeeper你需要配置文件。下面是一个示例，请在conf/zoo.cfg:

这个文件可以被称为任何东西，但为了这次讨论调用它 conf/zoo.cfg。更改dataDir的值，以指定现有目录(empty to start with)。以下是每个字段的含义:

# tickTime

ZooKeeper以毫秒作为基本时间单位。它用于执行心跳，最小会话超时将是tickTime的两倍。

# dataDir

存储内存中数据库快照的位置，除非另有指定否则将更新数据库的事务日志。

# clientPort

监听客户端连接的端口

现在你已经创建了配置文件，你可以启动ZooKeeper:

bin/zkServer.sh start

ZooKeeper使用log4j记录消息----在程序猿指南的[日志记录](https://zookeeper.apache.org/doc/current/zookeeperProgrammers.html#Logging)部分提供了更多详细信息。根据log4j配置，你将看到控制台(默认)和/或日志文件的日志消息。

这里概述的步骤运行ZooKeeper在但接单模型。没有复制，所以如果ZooKeeper处理失败，服务将会关闭，这对于多数开发情况都很好，但要在复制模式下运行ZooKeeper，请参见Running Replicated ZooKeeper。

## ZooKeeper存储库

对于长时间运行的生产系统，ZooKeeper必须外部管理(dataDir和日志)。有关详细信息，请参考 [维护](https://zookeeper.apache.org/doc/current/zookeeperAdmin.html#sc_maintenance) 部分。

## 1.5 连接 ZooKeeper

$ bin/zkCli.sh -server 127.0.0.1:2181

这使你可以执行简单的、类似文件的操作。

一旦连接成功，你应该会看到如下内容:

Connecting to localhost:2181

log4j:WARN No appenders could be found for logger (org.apache.zookeeper.ZooKeeper). log4j:WARN Please initialize the log4j system properly.

Welcome to ZooKeeper! JLine support is enabled [zkshell: 0]

从shell中键入帮助获取可从客户端执行的命令列表，如下:

[zkshell: 0] help

ZooKeeper host:port cmd args get path [watch]

ls path [watch]

set path data [version] delquota [-n|-b] path quit

printwatches on|off createpath data acl stat path [watch] listquota path history

setAcl path acl

getAcl path sync path redo cmdno

addauth scheme auth delete path [version] setquota -n|-b val path

从这里开始，可以尝试几个简单的命令感受一下这个简单的命令行界面。首先，从发出list命令开始，如在ls中一样:

[zkshell: 8] ls / [zookeeper]

下一步，通过运行创建新的znode，命令: create /zk\_test my\_data。这将创建一个新的znode，并将字符串”my\_data”与该节点关联。你应该看到:

[zkshell: 9] create /zk\_test my\_data Created /zk\_test

ls 的另一个问题 ls / 命令查看目录的外观::

[zkshell: 11] ls / [zookeeper, zk\_test]

请注意，现在已创建zk\_test目录

下一步，通过运行验证数据是否与znode关联 get 命令，如下所示:

[zkshell: 12] get /zk\_test my\_data

cZxid = 5

ctime = Fri Jun 05 13:57:06 PDT 2009

mZxid = 5

mtime = Fri Jun 05 13:57:06 PDT 2009

pZxid = 5

cversion = 0

dataVersion = 0

aclVersion = 0

ephemeralOwner = 0

dataLength = 7

numChildren = 0

我们可以通过 set 命令更改zk\_test 关联的数据，如下所示:

[zkshell: 14] set /zk\_test junk cZxid = 5

ctime = Fri Jun 05 13:57:06 PDT 2009

mZxid = 6

mtime = Fri Jun 05 14:01:52 PDT 2009

pZxid = 5

cversion = 0

dataVersion = 1

aclVersion = 0

ephemeralOwner = 0

dataLength = 4

numChildren = 0

[zkshell: 15] get /zk\_test junk

cZxid = 5

ctime = Fri Jun 05 13:57:06 PDT 2009

mZxid = 6

mtime = Fri Jun 05 14:01:52 PDT 2009

pZxid = 5

cversion = 0

dataVersion = 1

aclVersion = 0

ephemeralOwner = 0

dataLength = 4

numChildren = 0

请注意，在设置数据之后，我们做了一个get，并且缺失发生了更改。最后让我们通过发出删除节点:

[zkshell: 16] delete /zk\_test [zkshell: 17] ls / [zookeeper]

[zkshell: 18]

现在就这样要了解更多内容，请继续本文档的其余部分，并查看[Programmer’s Guide](https://zookeeper.apache.org/doc/current/zookeeperProgrammers.html)。.

## 1.6 用 ZooKeeper编程

ZooKeeper有一个Java和C绑定。他们在功能上是等同的。C绑定存在于两种变体中: 单线程和多线程。这些仅在消息传递循环的执行方式上有所不同。有关更多信息，请参考[ZooKeeper程序猿指南](https://zookeeper.apache.org/doc/current/zookeeperProgrammers.html#ch_programStructureWithExample)中有关使用不同api的实例代码的编程示例。

## 运行赋值的ZooKeeper

在单节点模式下运行ZooKeeper，便于评估、开发和测试。但是在生产中，你应该在赋值模式下运行ZooKeeper。同一应用程序中的一组已赋值的服务器成为仲裁，在赋值模式下，仲裁中的所有服务器都有相同配置文件的副本。

Note:

对于复制模式，至少需要三台服务器，强烈建议你有一个技术的服务器。如果你只有两台服务器，那么你正处于一种情况，如果其中一个失败，则没有足够的机器形成节点多数仲裁模式。两台服务器本质上不如一台服务器稳定，因为有两个单点故障。.

赋值模式所需的 conf/zoo.cfg 文件类似于在独立模式下使用，但存在一些差异。下面是一个示例:

tickTime=2000 dataDir=/var/lib/zookeeper clientPort=2181 initLimit=5

syncLimit=2 server.1=zoo1:2888:3888 server.2=zoo2:2888:3888 server.3=zoo3:2888:3888

新条目initLimit是ZooKeeper使用的超时限制，ZooKeeper必须连接到一个Leader的时间长度。条目syncLimit限制了服务器从引线到日期的距离。.

通过这两个超时，你可以使用tickTime指定时间单位。在本例中，initLimit的超时时间为5个，在2000milleseconds，一个刻度，或者10秒。

窗体服务器的项。X列出构成ZooKeeper服务的服务器。当服务器启动时，它通过在数据目录中查找文件myid指导他是哪个服务器。该文件在ASCII中包括服务号。

最后，请注意每个服务器名称后的两个端口号:”2888”和“3888”。对等方使用前一个端口连接到其他对等方。这样的连接是必要的，以便对等方可以进行通信，例如，同意更新的顺序。更具体地说，ZooKeeper使用此端口将follow连接到leader。当一个新的leader出现，一个follow打开一个TCP连接到leader使用这个端口。因为默认的leader选举也使用TCP，我们目前需要另一个port来进行leader选举。这是服务器选项中的第二个端口。

如果要在单台机器上测试多台服务器，请将服务器名称指定为具有唯一仲裁和leader选举端口的本地主机(即上述实例中的2888:3888,2889:3889,2890:3890)配置文件。当然，单独的dataDirs和不同的clientPort也是必要的(在上面赋值的例子汇总，在一个本地主机上运行，你让然有三个配置文件)。请注意，在一台机器上设置多个服务器不会产生任何冗余。如果发生了导致机器死机的事情，所有的ZooKeeper服务器都将处于脱机状态。完全冗余要求每台服务器都有自己的机器。同一物理主机上的多个虚拟机仍然容易出现该主机的完全故障。

## 1.8 其他优化

还有几个其他配置参数可以大大提高性能:

* 若要在更新时获得低延迟，还有一个专用事务日志目录很重要。默认情况下，事务日志与数据快照和myid文件方在同一个目录中
* *[待定:其他配置参数是什么？]*