**目录**

[1. 概述 1](#_Toc453753316)

[2. 普通主从模型 2](#_Toc453753317)

[2.1准备1个Master 3](#_Toc453753318)

[2.2准备1个Slave 4](#_Toc453753319)

[2.3操作Master观察Slave 5](#_Toc453753320)

[2.4运行后再添加一台从库 7](#_Toc453753321)

[2.5再说两句 9](#_Toc453753322)

[2.5.1Master-Slave读写属性 9](#_Toc453753323)

[2.5.2Slave如何接替Master的工作（已测） 9](#_Toc453753324)

[2.5.3切换Master-Slave（未测） 10](#_Toc453753325)

[2.5.4更新Master 10](#_Toc453753326)

[2.6解决主从不同步 10](#_Toc453753327)

[3. 副本集 12](#_Toc453753328)

[3.1 启动两个结点 13](#_Toc453753329)

[3.2初始化两个副本集结点 14](#_Toc453753330)

[3.3设置仲裁结点 14](#_Toc453753331)

[3.4关掉一个副本结点，看看主节点如何变化 17](#_Toc453753332)

[3.5副本结点不具有读文档的权限，需要设置slaveOk=true才行， 19](#_Toc453753333)

[3.6搭建一个没有仲裁结点的副本集群 20](#_Toc453753334)

[3.7再说点什么 26](#_Toc453753335)

[3.7.1其他节点类型 26](#_Toc453753336)

[3.7.2副本集服务访问 27](#_Toc453753337)

[3.7.3更改结点优先级 27](#_Toc453753338)

[3.7.4添加副本集成员 27](#_Toc453753339)

[3.7.5删除副本集成员 32](#_Toc453753340)

[4. 参考资料 32](#_Toc453753341)

**jar cvfm MyWordCount.jar manifest.data \*.class**

# 概述

主从复制常用于数据备份容灾、数据恢复、读写分离（提高性能）。一个简单的主从模型如下：



# 普通主从模型

普通主从模型：



迫于机器数量，我们没有用多个机器来搭建。但是MongoDB可以在单机上不同目录部署多台，这个，下面我们在一台机器上做一个简单的实践。

主从复制可用选项如下：

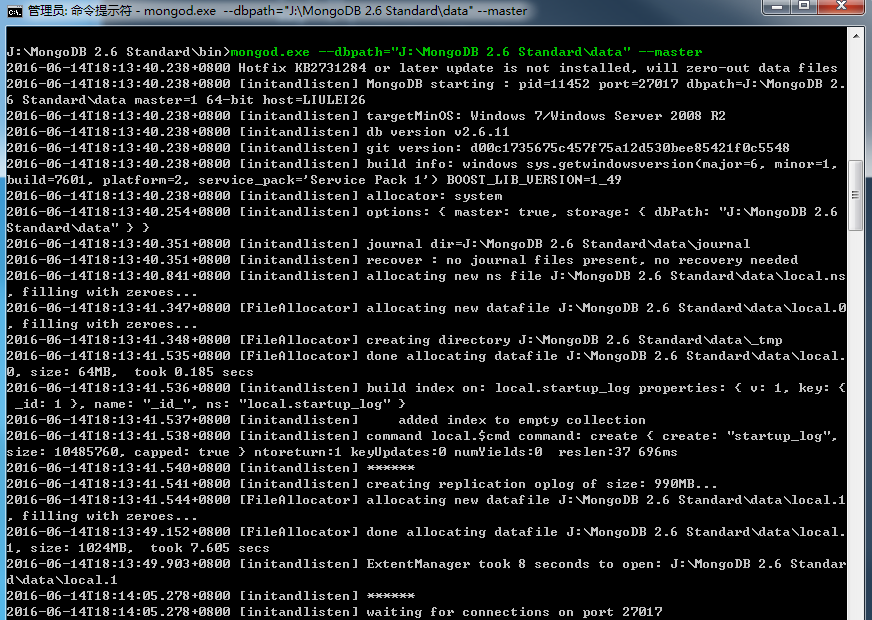
|  |  |
| --- | --- |
| 参数 | 描述 |
| --only | 在从节点上指定只复制特定的某个数据库(默认是复制所有数据库) |
| --slavedelay | 用在从节点上，当应用主节点的操作时，从节点增加延时复制(单位秒).这样就能轻松设置延时从节点，这种节点对用户无意中删除重要文档或者插入垃圾数据等有防护作用，这些不良操作都会被复制到所有的从节点上，通过延时执行操作，可以有个恢复的时间差。 |
| --fastsync | 以主节点的数据快照为基础启动从节点.如果数据目录一开始是主节点的数据快照，从节点用这个选项启动要比做完整的同步快的多. |
| --autoresync | 如果从节点与主节点不同步了，则自动重新同步. |
| --oplogsize | 主节点oplog的大小(单位MB). |

## 2.1准备1个Master

我们在J:\MongoDB 2.6 Standard放了一个MongoDB结点，我们如下启动：

**mongod.exe --dbpath="J:\MongoDB 2.6 Standard\data" --master**

使用默认端口27017，**注意J:\MongoDB 2.6 Standard\data要存在**。启动之后可以看到如下信息：



--master就是将其指定位Master库。

## 2.2准备1个Slave

我们在J:\MongoDB 2.6 Standard-Slave1也放一个MongoDB结点，启动如下：

**mongod.exe --dbpath="J:\MongoDB 2.6 Standard-Slave1\data" --port=8888 --slave --source=127.0.0.1:27017**

启动之后，在Master端我们可以看到从库的连接信息。

--port：指定启动端口，因为我们在同一台机器上模拟，所以从库必须更换端口；

--slave：指定该库是从库；

--source：指定主库ip:port信息。

## 2.3操作Master观察Slave

我们连接Master库，添加一些数据：

**C:\Users\Administrator>mongo**

**2016-06-14T18:24:37.541+0800 Hotfix KB2731284 or later update is not installed， will zero-out data files**

**MongoDB shell version: 2.6.11**

**connecting to: test**

**> show dbs;**

**admin (empty)**

**local 1.078GB**

**> use student;**

**switched to db student**

**> db.class\_info.insert({"name":"zhangsan"， "stu\_no":"00000001"});**

**WriteResult({ "nInserted" : 1 })**

**> db.class\_info.find();**

**{ "\_id" : ObjectId("575fdbd42d63c188ef1eea6e")， "name" : "zhangsan"， "stu\_no" : "00000001" }**

**>**

然后，我们连接Slave从库查看一下：

**C:\Users\Administrator>mongo 127.0.0.1:8888**

**2016-06-14T18:26:53.531+0800 Hotfix KB2731284 or later update is not installed， will zero-out data files**

**MongoDB shell version: 2.6.11**

**connecting to: 127.0.0.1:8888/test**

**> show dbs;**

**admin (empty)**

**local 0.078GB**

**student 0.078GB**

**> use student;**

**switched to db student**

**> show collections;**

**class\_info**

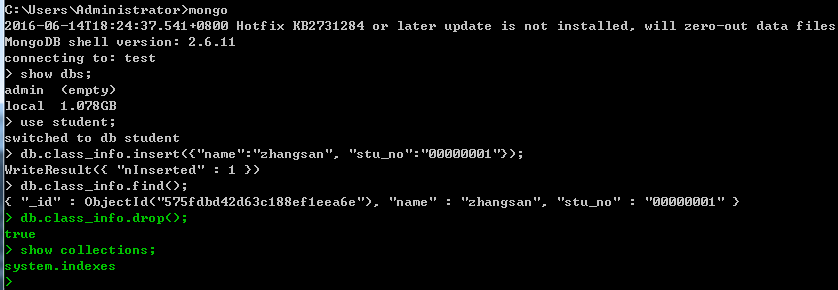
**system.indexes**

**> db.class\_info.find();**

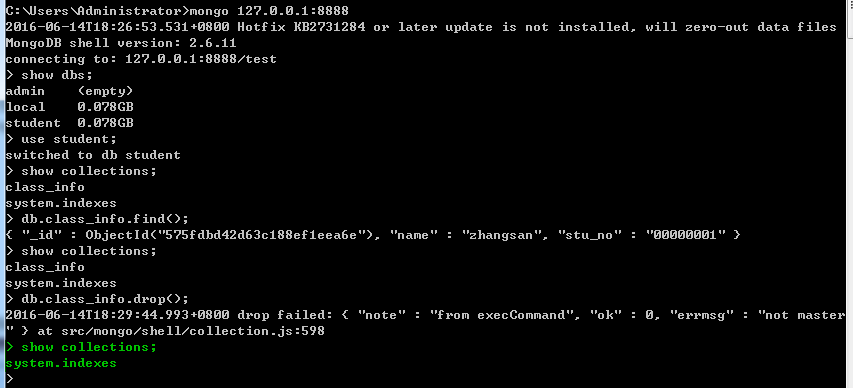
**{ "\_id" : ObjectId("575fdbd42d63c188ef1eea6e")， "name" : "zhangsan"， "stu\_no" : "00000001" }**

**>**

我们看到信息同步到从库了，下面我们在主库删除class\_info这个文档。



查看从库如下：

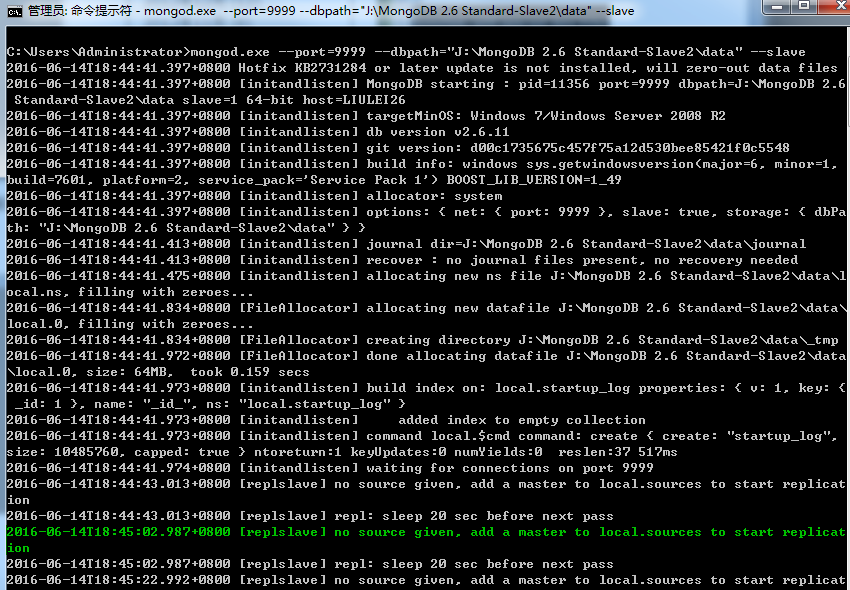


## 2.4运行后再添加一台从库

我们在目录J:\MongoDB 2.6 Standard-Slave2放置一个MongoDB结点，

**mongod.exe --port=9999 --dbpath="J:\MongoDB 2.6 Standard-Slave2\data" --slave**

启动之后会如下提示我们没有找到Master结点：



下面我们连接这个库，添加主库信息：

**J:\MongoDB 2.6 Standard-Slave2\data>mongo 127.0.0.1:9999**

**2016-06-14T18:47:02.568+0800 Hotfix KB2731284 or later update is not installed， will zero-out data files**

**MongoDB shell version: 2.6.11**

**connecting to: 127.0.0.1:9999/test**

**> use local;**

**switched to db local**

**> show collections;**

**startup\_log**

**system.indexes**

**> db.sources.insert({"host":"127.0.0.1:27017"});**

**WriteResult({ "nInserted" : 1 })**

**> db.sources.find();**

**{ "\_id" : ObjectId("575fe112d721dd621aabd999")， "host" : "127.0.0.1:27017"， "source" : "main"， "syncedTo"**

**: Timestamp(1465901339， 1) }**

**> show dbs;**

**admin (empty)**

**local 0.078GB**

**student (empty)**

**>**

在从库数据库local中，sources记录了主库的信息，我们将其添加即可。如上，添加完成后，主库的信息也就同步到从库中来了。

## 2.5再说两句

### 2.5.1Master-Slave读写属性

Master具有操作数据的所有权限；

Slave没有更新数据的权限，可以读数据。

### 2.5.2Slave如何接替Master的工作（已测）

如果Master坏掉了，那么Slave就需要接替Master的工作，通过如下步骤进行：

1. 停止Slave上面的mongod进程；
2. 删除Slave数据目录下的local.\*所有文件；
3. 以master方式再次启动Slave；
4. 更新其他Slave中的source地址；

### 2.5.3切换Master-Slave（未测）

假设已经具备主机A和从机B，此时想切换它们的角色，步骤如下：（假设A是健康的）

1. 用fsync命令暂停A上的写操作，
2. 确定B是从机，关闭B上的服务
3. 清空B上的local.\*文件
4. 用-master选项重启B服务
5. 在B上执行一次写操作，初始化oplog，获得一个同步起始点
6. 关闭B服务，此时B已经有了信的local.\*文件
7. 关闭A服务，并且用B上新的local.\*文件来代替A上的local.\*文件（拷贝之前，记得先压缩，因为文件可能很大）
8. 用-master选项重启B服务
9. 在平时的slave选项上加一个-fastsync选项来重启A服务。

如果A不是健康的，但是硬件是健康的，那么跳过上面的前两步，并且用B上所有文件去替换A上的文件，重启服务。

### 2.5.4更新Master（已测）

（1）重启Slave的mongod，**不加--slave、--source选项**。

（2）进入重启之后的Slave，执行如下命令：

**student (empty)**

**use local;**

**db.sources.update({host : "132.129.31.213：10000"}, {$set : {host : "132.129.31.213：10001"}});**

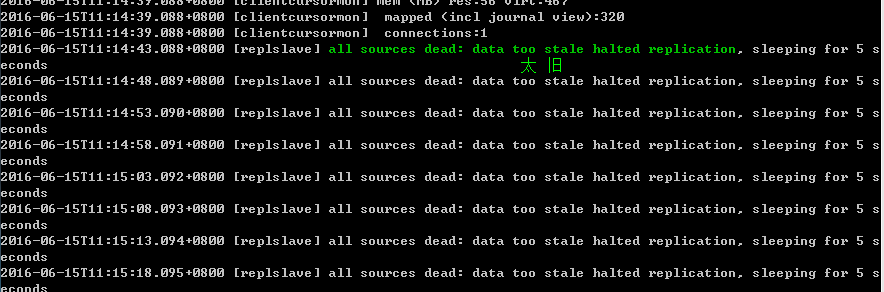
**>**

（3）重启Slave上的mongod，**加上--slave、--source选项**；

更新之后有可能出现主从不同步的情况，解决方法参考2.6。

### 2.6解决主从不同步

如果更改Master重启后从库数据无法更新，一直提示“2016-06-15T11:14:48.089+0800 [replslave] all sources dead: data too stale halted replication, sleeping for 5 seconds”如下：



提示数据太陈旧，此时我们需要手动将主库同步到从库中（通常是这样，但是你发现从库数据完成，而主库数据有问题，那么可能需要将从库数据同步到主库中，做法类似，将主从数据库停止后，将主从关系颠倒，然后再手动同步）。我们进入从库中：

**> use admin;**

**switched to db admin**

**> db.runCommand({"resync":1})**

**{ "info" : "triggered resync for all sources", "ok" : 1 }**

**> use stu;**

**switched to db stu**

**> db.class\_info.find();**

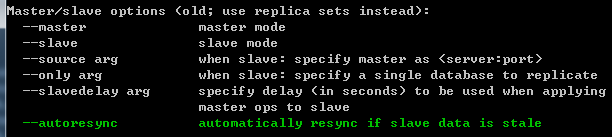
**{ "\_id" : ObjectId("5760c257d824b2c520822574"), "insert\_time" : "1111" }**

**{ "\_id" : ObjectId("5760c46042927e03d0e9a57b"), "insert\_time" : "2222" }**

**{ "\_id" : ObjectId("5760c56f42927e03d0e9a57c"), "insert\_time" : "3333" }**

**{ "\_id" : ObjectId("5760c62942927e03d0e9a57d"), "insert\_time" : "4444" }**

或者直接在从库启动的时候，加上选项--autoresync=true也可自动将主库数据同步到从库中来。



# 副本集

两个典型的副本集模型：

带有仲裁结点：



不带仲裁结点：



PS：注意这里的数据同步包括：操作数据的同步、选举投票心跳数据同步等。

副本集与普通的主从模型相比有如下特点：

1. 没有固定的主数据库；
2. 某个数据库宕机了，集群会选出另一个数据库作为主数据库，这样就具备了自动故障恢复功能。

下面来建立一个副本集群：

## 3.1 启动两个结点

我们在目录J:\MongoDB 2.6 Standard-Slave1、J:\MongoDB 2.6 Standard-Slave2下分别放置MongoDB结点。下面我们依次启动这两个结点：

1. 启动第一个节点

**mongod --dbpath="J:\MongoDB 2.6 Standard-Slave1\data" --port=2222 --replSet lt/127.0.0.1:3333**

--port：自己使用的端口；

--replSet：lt是副本集名称，3333是另一个副本使用的端口。

只有一个结点启动时，会出现对方积极拒绝连接的错误，这个没有关系，另一个结点启动就OK了。

（2）启动第二个结点

**mongod --dbpath="J:\MongoDB 2.6 Standard-Slave2\data" --port=3333 --replSet lt/127.0.0.1:2222**

## 3.2初始化两个副本集结点

任意进入其中一个结点，必须在到admin数据库中：

**db.runCommand({"replSetInitiate":{"\_id":"lt", "members":[{"\_id":1, "host":"127.0.0.1:2222"}, {"\_id":2, "host":"127.0.0.1:3333"}]}});**

初始化完成后，就出现了主节点和副本结点，这个类似于普通主从，主节点CURD数据，从节点不能修改数据。

## 3.3设置仲裁结点

通常副本集的个数设置为奇数个，这样便于选出主节点，但是如果剩下了偶数个结点，由于存在选举不超过半数的的情况，这样就会撕逼。仲裁结点就是解决这种撕逼问题的。

我们在目录J:\MongoDB 2.6 Standard下放置一个MongoDB用来做仲裁结点，我们如下启动这个结点：

**mongod --dbpath="J:\MongoDB 2.6 Standard\data" --port 4444 --replSet lt/127.0.0.1:2222**

然后把自己作为仲裁结点加入lt副本集群中：

**rs.addArb("127.0.0.1:4444")**

查看集群状态：

**lt:PRIMARY> rs.status();**

**{**

**"set" : "lt",**

**"date" : ISODate("2016-06-14T13:05:56Z"),**

**"myState" : 1,**

**"members" : [**

**{**

**"\_id" : 1,**

**"name" : "127.0.0.1:2222",**

**"health" : 1,**

**"state" : 1,**

**"stateStr" : "PRIMARY",**

**"uptime" : 332,**

**"optime" : Timestamp(1465909505, 1),**

**"optimeDate" : ISODate("2016-06-14T13:05:05Z"),**

**"electionTime" : Timestamp(1465909307, 1),**

**"electionDate" : ISODate("2016-06-14T13:01:47Z"),**

**"self" : true**

**},**

**{**

**"\_id" : 2,**

**"name" : "127.0.0.1:3333",**

**"health" : 1,**

**"state" : 2,**

**"stateStr" : "SECONDARY",**

**"uptime" : 255,**

**"optime" : Timestamp(1465909505, 1),**

**"optimeDate" : ISODate("2016-06-14T13:05:05Z"),**

**"lastHeartbeat" : ISODate("2016-06-14T13:05:55Z"),**

**"lastHeartbeatRecv" : ISODate("2016-06-14T13:05:55Z"),**

**"pingMs" : 0,**

**"syncingTo" : "127.0.0.1:2222"**

**},**

**{**

**"\_id" : 3,**

**"name" : "127.0.0.1:4444",**

**"health" : 1,**

**"state" : 7,**

**"stateStr" : "ARBITER",**

**"uptime" : 51,**

**"lastHeartbeat" : ISODate("2016-06-14T13:05:55Z"),**

**"lastHeartbeatRecv" : ISODate("2016-06-14T13:05:56Z"),**

**"pingMs" : 0**

**}**

**],**

**"ok" : 1**

**}**

## 3.4关掉一个副本结点，看看主节点如何变化

目前的主节点是2222，我们将其关闭，然后查看状态：

**lt:PRIMARY> rs.status();**

**{**

**"set" : "lt",**

**"date" : ISODate("2016-06-14T13:11:04Z"),**

**"myState" : 1,**

**"members" : [**

**{**

**"\_id" : 1,**

**"name" : "127.0.0.1:2222",**

**"health" : 0,**

**"state" : 8,**

**"stateStr" : "(not reachable/healthy)",**

**"uptime" : 0,**

**"optime" : Timestamp(1465909505, 1),**

**"optimeDate" : ISODate("2016-06-14T13:05:05Z"),**

**"lastHeartbeat" : ISODate("2016-06-14T13:11:02Z"),**

**"lastHeartbeatRecv" : ISODate("2016-06-14T13:10:33Z"),**

**"pingMs" : 0**

**},**

**{**

**"\_id" : 2,**

**"name" : "127.0.0.1:3333",**

**"health" : 1,**

**"state" : 1,**

**"stateStr" : "PRIMARY",**

**"uptime" : 849,**

**"optime" : Timestamp(1465909505, 1),**

**"optimeDate" : ISODate("2016-06-14T13:05:05Z"),**

**"electionTime" : Timestamp(1465909841, 1),**

**"electionDate" : ISODate("2016-06-14T13:10:41Z"),**

**"self" : true**

**},**

**{**

**"\_id" : 3,**

**"name" : "127.0.0.1:4444",**

**"health" : 1,**

**"state" : 7,**

**"stateStr" : "ARBITER",**

**"uptime" : 357,**

**"lastHeartbeat" : ISODate("2016-06-14T13:11:04Z"),**

**"lastHeartbeatRecv" : ISODate("2016-06-14T13:11:02Z"),**

**"pingMs" : 0**

**}**

**],**

**"ok" : 1**

**}**

## 3.5副本结点不具有读文档的权限，需要设置slaveOk=true才行，

执行如下命令：

**lt:SECONDARY> show dbs;**

**admin (empty)**

**local 1.078GB**

**test 0.078GB**

**lt:SECONDARY> use test;**

**switched to db test**

**lt:SECONDARY> show collections;**

**2016-06-15T08:49:14.525+0800 error: { "$err" : "not master and slaveOk=false", "code" : 13435 } at src/mo**

**ngo/shell/query.js:131**

**lt:SECONDARY> db.getMongo().setSlaveOk();**

**lt:SECONDARY> show collections;**

**stu**

**system.indexes**

**lt:SECONDARY> db.stu.find();**

**{ "\_id" : ObjectId("576002f0d65dc75ab6a63a5c"), "zhangsan" : "0000" }**

**lt:SECONDARY>**

设置之后，Secondary结点就具有了读数据权限，但依然不能写数据。

## 3.6搭建一个没有仲裁结点的副本集群

注意：搭建没有仲裁结点的集群通常建议是奇数个结点，在此只是做一个实验用了4个结点。

3.6.1准备4个MongoDB结点

在目录J:\MongoDB 2.6 Standard、J:\MongoDB 2.6 Standard-1、J:\MongoDB 2.6 Standard-2、J:\MongoDB 2.6 Standard-3下面各准备一个结点。然后以如下方式启动：

**mongod.exe --port 10000 --dbpath="J:\MongoDB 2.6 Standard\data" --replSet lt**

**mongod.exe --port 10001 --dbpath="J:\MongoDB 2.6 Standard-1\data" --replSet lt**

**mongod.exe --port 10002 --dbpath="J:\MongoDB 2.6 Standard-2\data" --replSet lt**

**mongod.exe --port 10004 --dbpath="J:\MongoDB 2.6 Standard-3\data" --replSet lt**

3.6.2初始化副本集

登录到任意一台机器上，按照如下命令执行：

**C:\Users\Administrator>mongo 127.0.0.1:10002**

**2016-06-15T09:04:22.919+0800 Hotfix KB2731284 or later update is not installed, will zero-out data files**

**MongoDB shell version: 2.6.11**

**connecting to: 127.0.0.1:10002/test**

**> use admin;**

**switched to db admin**

**> config = {\_id:"lt", members:[{\_id:0, host:"127.0.0.1:10000"}, {\_id:1, host:"127.0.0.1:10001"}, {\_id:2, host:"127.0.0.1:10002"}, {\_id:3, host:"127.0.0.1:10004"}]};**

**{**

**"\_id" : "lt",**

**"members" : [**

**{**

**"\_id" : 0,**

**"host" : "127.0.0.1:10000"**

**},**

**{**

**"\_id" : 1,**

**"host" : "127.0.0.1:10001"**

**},**

**{**

**"\_id" : 2,**

**"host" : "127.0.0.1:10002"**

**},**

**{**

**"\_id" : 3,**

**"host" : "127.0.0.1:10004"**

**}**

**]**

**}**

**> rs.initiate(config);**

**{**

**"info" : "Config now saved locally. Should come online in about a minute.",**

**"ok" : 1**

**}**

**lt:PRIMARY> rs.status();**

**{**

**"set" : "lt",**

**"date" : ISODate("2016-06-15T01:09:23Z"),**

**"myState" : 1,**

**"members" : [**

**{**

**"\_id" : 0,**

**"name" : "127.0.0.1:10000",**

**"health" : 1,**

**"state" : 2,**

**"stateStr" : "SECONDARY",**

**"uptime" : 87,**

**"optime" : Timestamp(1465952876, 1),**

**"optimeDate" : ISODate("2016-06-15T01:07:56Z"),**

**"lastHeartbeat" : ISODate("2016-06-15T01:09:22Z"),**

**"lastHeartbeatRecv" : ISODate("2016-06-15T01:09:23Z"),**

**"pingMs" : 0,**

**"syncingTo" : "127.0.0.1:10002"**

**},**

**{**

**"\_id" : 1,**

**"name" : "127.0.0.1:10001",**

**"health" : 1,**

**"state" : 2,**

**"stateStr" : "SECONDARY",**

**"uptime" : 87,**

**"optime" : Timestamp(1465952876, 1),**

**"optimeDate" : ISODate("2016-06-15T01:07:56Z"),**

**"lastHeartbeat" : ISODate("2016-06-15T01:09:22Z"),**

**"lastHeartbeatRecv" : ISODate("2016-06-15T01:09:23Z"),**

**"pingMs" : 0,**

**"syncingTo" : "127.0.0.1:10002"**

**},**

**{**

**"\_id" : 2,**

**"name" : "127.0.0.1:10002",**

**"health" : 1,**

**"state" : 1,**

**"stateStr" : "PRIMARY",**

**"uptime" : 347,**

**"optime" : Timestamp(1465952876, 1),**

**"optimeDate" : ISODate("2016-06-15T01:07:56Z"),**

**"electionTime" : Timestamp(1465952896, 1),**

**"electionDate" : ISODate("2016-06-15T01:08:16Z"),**

**"self" : true**

**},**

**{**

**"\_id" : 3,**

**"name" : "127.0.0.1:10004",**

**"health" : 1,**

**"state" : 2,**

**"stateStr" : "SECONDARY",**

**"uptime" : 85,**

**"optime" : Timestamp(1465952876, 1),**

**"optimeDate" : ISODate("2016-06-15T01:07:56Z"),**

**"lastHeartbeat" : ISODate("2016-06-15T01:09:23Z"),**

**"lastHeartbeatRecv" : ISODate("2016-06-15T01:09:23Z"),**

**"pingMs" : 0,**

**"syncingTo" : "127.0.0.1:10002"**

**}**

**],**

**"ok" : 1**

**}**

好，至此只有副本集没有仲裁节点的副本集群我们就搭建完成了。也可以通过该种方式做一个带有仲裁结点集群，配置如下：

**config = {\_id:"lt", members:[{\_id:0, host:"127.0.0.1:10000"}, {\_id:1, host:"127.0.0.1:10001", arbiterOnly:true}, {\_id:2, host:"127.0.0.1:10002"}, {\_id:3, host:"127.0.0.1:10004"}]};**

**rs.initiate(config);**

## 3.7再说点什么

### 3.7.1其他节点类型

除了**主节点、副本结点、仲裁结点**之外，还有**Secondary-Only**（不能成为Primary结点，只作为副本结点，防止一些性能不高的结点作为主节点）、**Hidden**（这类节点是不能够被客户端制定IP引用，也不能被设置为主节点，但是可以投票，一般用于备份数据）、**Delayed**（可以指定一个时间延迟从primary节点同步数据。主要用于备份数据，如果实时同步，误删除数据马上同步到从节点，恢复又恢复不了）、**Non-Voting**（没有选举权的Secondary结点，纯粹用于数据备份）。

### 3.7.2副本集服务访问

副本集这种方式在Primary发生故障时，可以自动从Secondary节点中选出一个作为Primary，这样非常好。

在客户端访问代码中，我们也可以一次性将集群中所有的IP:Port配置进去，当初次访问的Primary故障时，程序自动尝试连接其他节点，这样客户端代码也就自动切换了。

### 3.7.3更改结点优先级

选举Primary结点可以根据优先来确定，设置各个节点的优先级如下（必须在当前Primary结点进行，否则出错）：

**rs.conf();**

**cfg=rs.conf();**

**cfg.members[0].priority=1**

**cfg.members[1].priority=1**

**cfg.members[2].priority=10**

**rs.reconfig(cfg);**

数字越大优先级越高，优先级为0的节点不能作为Primary。

### 3.7.4添加副本集成员

**lt:PRIMARY> rs.add("127.0.0.1:10005")**

**{ "ok" : 1 }**

**lt:PRIMARY> rs.status();**

**{**

**"set" : "lt",**

**"date" : ISODate("2016-06-15T01:51:58Z"),**

**"myState" : 1,**

**"members" : [**

**{**

**"\_id" : 0,**

**"name" : "127.0.0.1:10000",**

**"health" : 1,**

**"state" : 2,**

**"stateStr" : "SECONDARY",**

**"uptime" : 2642,**

**"optime" : Timestamp(1465955458, 1),**

**"optimeDate" : ISODate("2016-06-15T01:50:58Z"),**

**"lastHeartbeat" : ISODate("2016-06-15T01:51:58Z"),**

**"lastHeartbeatRecv" : ISODate("2016-06-15T01:51:56Z"),**

**"pingMs" : 0,**

**"syncingTo" : "127.0.0.1:10002"**

**},**

**{**

**"\_id" : 1,**

**"name" : "127.0.0.1:10001",**

**"health" : 1,**

**"state" : 2,**

**"stateStr" : "SECONDARY",**

**"uptime" : 2642,**

**"optime" : Timestamp(1465955458, 1),**

**"optimeDate" : ISODate("2016-06-15T01:50:58Z"),**

**"lastHeartbeat" : ISODate("2016-06-15T01:51:58Z"),**

**"lastHeartbeatRecv" : ISODate("2016-06-15T01:51:56Z"),**

**"pingMs" : 0,**

**"syncingTo" : "127.0.0.1:10002"**

**},**

**{**

**"\_id" : 2,**

**"name" : "127.0.0.1:10002",**

**"health" : 1,**

**"state" : 1,**

**"stateStr" : "PRIMARY",**

**"uptime" : 2902,**

**"optime" : Timestamp(1465955458, 1),**

**"optimeDate" : ISODate("2016-06-15T01:50:58Z"),**

**"electionTime" : Timestamp(1465952896, 1),**

**"electionDate" : ISODate("2016-06-15T01:08:16Z"),**

**"self" : true**

**},**

**{**

**"\_id" : 3,**

**"name" : "127.0.0.1:10004",**

**"health" : 1,**

**"state" : 2,**

**"stateStr" : "SECONDARY",**

**"uptime" : 2640,**

**"optime" : Timestamp(1465955458, 1),**

**"optimeDate" : ISODate("2016-06-15T01:50:58Z"),**

**"lastHeartbeat" : ISODate("2016-06-15T01:51:57Z"),**

**"lastHeartbeatRecv" : ISODate("2016-06-15T01:51:57Z"),**

**"pingMs" : 0,**

**"syncingTo" : "127.0.0.1:10002"**

**},**

**{**

**"\_id" : 4,**

**"name" : "127.0.0.1:10005",**

**"health" : 1,**

**"state" : 2,**

**"stateStr" : "SECONDARY",**

**"uptime" : 60,**

**"optime" : Timestamp(1465955458, 1),**

**"optimeDate" : ISODate("2016-06-15T01:50:58Z"),**

**"lastHeartbeat" : ISODate("2016-06-15T01:51:58Z"),**

**"lastHeartbeatRecv" : ISODate("2016-06-15T01:51:58Z"),**

**"pingMs" : 0,**

**"syncingTo" : "127.0.0.1:10002"**

**}**

**],**

**"ok" : 1**

**}**

**lt:PRIMARY>**

同样的，这些配置必须在Primary结点上执行，添加之后数据会自动同步到新添加的成员中。

### 3.7.5删除副本集成员

rs.remove(“ip:port”);

# 参考资料

1. http://www.cnblogs.com/huangxincheng/archive/2012/03/04/2379755.html
2. http://www.cnblogs.com/refactor/archive/2012/08/13/2600140.html
3. http://www.zhihu.com/question/27648448
4. http://my.oschina.net/willSoft/blog/34993
5. https://docs.mongodb.com/manual/reference/method/Mongo.setSlaveOk/
6. http://www.csdn.net/article/2014-04-09/2819221-build-high-avialable-mongodb-cluster-part-1/2
7. http://blog.itpub.net/29254281/viewspace-1176553/
8. http://blog.51yip.com/nosql/1582.html
9. http://blog.sina.com.cn/s/blog\_62ae07b10101hcly.html
10. http://elain.blog.51cto.com/3339379/760266/