# single启动

## 启动命令:

mongod --dbpath <path for store data>

# 搭建副本集

## 1.准备三台机器，分别安装好mongodb。

## 2.启动每台mongodb

命令：mongod  --dbpath /data/mongodbtest/replset/data   --replSet testrepl

参数分别是

数据存储位置（自己任意指定路径）：/data/mongodbtest/replset/data

副本集参数名称（可任意命名）：testrepl

## 3.配置副本集

任意登录一台mongodb数据库

#使用admin数据库

use admin

定义副本集配置变量，这里的 \_id:” testrepl” 和上面命令参数“ –replSet testrepl” 要保持一样。

config = { \_id:" testrepl ", members:[

{\_id:0,host:" 192.168.1.136:27017 ",priority:2},

{\_id:1,host:" 192.168.1.137:27017 ",priority:1},

{\_id:2,host:" 192.168.1.138:27017 ",arbiterOnly:true}]

}

使用副本集配置初始化副本集

rs.initiate(config);

接下来需等待一段时间。

完成后可查看节点状态：

rs.state();

输出：

{

         "set" : "tesrepl",

          "date" : ISODate("2013-12-29T12:54:25Z"),

          "myState" : 1,

         "members" : [

                 {

                         "\_id" : 0,

                         "name" : "192.168.1.136:27017",

                         "health" : 1,

                         "state" : 2,

                         "stateStr" : "SECONDARY",

                         "uptime" : 1682,

                         "optime" : Timestamp(1388319973, 1),

                         "optimeDate" : ISODate("2013-12-29T12:26:13Z"),

                         "lastHeartbeat" : ISODate("2013-12-29T12:54:25Z"),

                         "lastHeartbeatRecv" : ISODate("2013-12-29T12:54:24Z"),

                         "pingMs" : 1,

                         "syncingTo" : "192.168.1.138:27017"

                 },

                 {

                         "\_id" : 1,

                         "name" : "192.168.1.137:27017",

                         "health" : 1,

                         "state" : 2,

                         "stateStr" : "SECONDARY",

                         "uptime" : 1682,

                         "optime" : Timestamp(1388319973, 1),

                         "optimeDate" : ISODate("2013-12-29T12:26:13Z"),

                         "lastHeartbeat" : ISODate("2013-12-29T12:54:25Z"),

                         "lastHeartbeatRecv" : ISODate("2013-12-29T12:54:24Z"),

                         "pingMs" : 1,

                         "syncingTo" : "192.168.1.138:27017"

                 },

                 {

                         "\_id" : 2,

                         "name" : "192.168.1.138:27017",

                         "health" : 1,

                         "state" : 1,

                         "stateStr" : "PRIMARY",

                         "uptime" : 2543,

                         "optime" : Timestamp(1388319973, 1),

                         "optimeDate" : ISODate("2013-12-29T12:26:13Z"),

                         "self" : true

                 }

         ],

         "ok" : 1

## 4.测试副本集数据复制功能

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | #在主节点192.168.1.138 上连接到终端：  mongo 127.0.0.1    #建立test 数据库。  use test;    往testdb表插入数据。  > db.testdb.insert({"test1":"testval1"})    #在副本节点 192.168.1.136、192.168.1.137 上连接到mongodb查看数据是否复制过来。  /data/mongodbtest/mongodb-linux-x86\_64-2.4.8/bin/mongo 192.168.1.136:27017    #使用test 数据库。  repset:SECONDARY> use test;    repset:SECONDARY> show tables; |

#输出

|  |  |  |
| --- | --- | --- |
| 1 | Sun Dec 29 21:50:48.590 error: { "$err" : "not master and slaveOk=false", "code" : 13435 } at src/mongo/shell/query.js:128 | |
| 1  2  3  4  5 | #mongodb默认是从主节点读写数据的，副本节点上不允许读，需要设置副本节点可以读。  repset:SECONDARY> db.getMongo().setSlaveOk();    #可以看到数据已经复制到了副本集。  repset:SECONDARY> db.testdb.find(); |

|  |  |
| --- | --- |
| 1  2 | #输出  { "\_id" : ObjectId("52c028460c7505626a93944f"), "test1" : "testval1" } |

## 5.测试副本集故障转移功能

先停掉主节点mongodb 138，查看136、137的日志可以看到经过一系列的投票选择操作，查看整个集群的状态，可以看到138为状态不可达。

|  |  |
| --- | --- |
| 1  2  3 | /data/mongodbtest/mongodb-linux-x86\_64-2.4.8/bin/mongo 192.168.1.136:27017    repset:SECONDARY> rs.status(); |

#输出

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48 | {          "set" : "testrepl",          "date" : ISODate("2013-12-29T14:28:35Z"),          "myState" : 2,          "syncingTo" : "192.168.1.137:27017",          "members" : [                  {                          "\_id" : 0,                          "name" : "192.168.1.136:27017",                          "health" : 1,                          "state" : 2,                          "stateStr" : "SECONDARY",                          "uptime" : 9072,                          "optime" : Timestamp(1388324934, 1),                          "optimeDate" : ISODate("2013-12-29T13:48:54Z"),                          "self" : true                  },                  {                          "\_id" : 1,                          "name" : "192.168.1.137:27017",                          "health" : 1,                          "state" : 1,                          "stateStr" : "PRIMARY",                          "uptime" : 7329,                          "optime" : Timestamp(1388324934, 1),                          "optimeDate" : ISODate("2013-12-29T13:48:54Z"),                          "lastHeartbeat" : ISODate("2013-12-29T14:28:34Z"),                          "lastHeartbeatRecv" : ISODate("2013-12-29T14:28:34Z"),                          "pingMs" : 1,                          "syncingTo" : "192.168.1.138:27017"                  },                  {                          "\_id" : 2,                          "name" : "192.168.1.138:27017",                          "health" : 0,                          "state" : 8,                          "stateStr" : "(not reachable/healthy)",                          "uptime" : 0,                          "optime" : Timestamp(1388324934, 1),                          "optimeDate" : ISODate("2013-12-29T13:48:54Z"),                          "lastHeartbeat" : ISODate("2013-12-29T14:28:35Z"),                          "lastHeartbeatRecv" : ISODate("2013-12-29T14:28:23Z"),                          "pingMs" : 0,                          "syncingTo" : "192.168.1.137:27017"                  }          ],          "ok" : 1  } |

再启动原来的主节点 138，发现138 变为 SECONDARY，还是137 为主节点 PRIMARY。

## 6.添加用户

现在需要创建一个帐号，该账号需要有grant权限，即：账号管理的授权权限。注意一点，帐号是跟着库走的，所以在指定库里授权，必须也在指定库里验证(auth)。

> use admin

> db.createUser({user: "dba",pwd: "dba", roles: [ { role: "userAdminAnyDatabase", db: "admin" } ] })

返回： Successfully added user: { "user" : "dba", "roles" : [ { "role" : "userAdminAnyDatabase", "db" : "admin" } ]}

刚建立了 userAdminAnyDatabase 角色，用来管理用户，可以通过这个角色来创建、删除用户。

验证：需要开启auth参数。

mongod--dbpath"D:\develop\mongodb\data"--logpath"D:\develop\mongodb\data\log\MongoDB.log" --auth

链接到mongodb

>show dbs

没有权限：返回

Error: listDatabases failed:{ "ok" : 0, "errmsg" : "not authorized on admin to execute command { listDatabases: 1.0 }", "code" : 13} at Error (<anonymous>) at Mongo.getDBs (src/mongo/shell/mongo.js:47:15) at shellHelper.show (src/mongo/shell/utils.js:630:33) at shellHelper (src/mongo/shell/utils.js:524:36) at (shellhelp2):1:1 at src/mongo/shell/mongo.js:47> use admin #验证，因为在admin下面添加的帐号，所以要到admin下面验证。

>use admin

>db.auth('dba','dba');

1

> show dbs;

admin 0.078GB

local 0.078GB

>use test #在test库里创建帐号

>db.createUser( { user: "zjy", pwd: "zjy", roles: [ { role: "readWrite", db: "test" }]}) ;

Successfully added user: { "user" : "zjy", "roles" : [ { "role" : "readWrite", #读写账号 "db" : "test" } ]}

## 7.验证权限

上面创建了2个帐号，现在验证下：验证前提需要一个集合

> db.abc.insert({"a":1,"b":2}) #插入失败，没有权限，userAdminAnyDatabase 权限只是针对用户管理的，对其他是没有权限的。

WriteResult({

"writeError" : {

"code" : 13,

"errmsg" : "not authorized on test to execute command { insert: \"abc\", documents: [ { \_id: ObjectId('55915185d629831d887ce2cb'), a: 1.0, b: 2.0 } ], ordered: true }"

}

})

root@zhoujinyi:/usr/local/mongo4# mongo --port=27020

MongoDB shell version: 3.0.4

connecting to: 127.0.0.1:27020/test

> use test

switched to db test

> db.auth('zjy','zjy') #用创建的readWrite帐号进行写入

1> db.abc.insert({"a":1,"b":2})

WriteResult({ "nInserted" : 1 })

> db.abc.insert({"a":11,"b":22})

WriteResult({ "nInserted" : 1 })

> db.abc.insert({"a":111,"b":222})

WriteResult({ "nInserted" : 1 })

> db.abc.find()

{ "\_id" : ObjectId("559151a1b78649ebd8316853"), "a" : 1, "b" : 2 }

{ "\_id" : ObjectId("559151cab78649ebd8316854"), "a" : 11, "b" : 22 }

{ "\_id" : ObjectId("559151ceb78649ebd8316855"), "a" : 111, "b" : 222 }

## 8.加验证启动副本集

>mongod --dbpath "d:\develop\mongodb\data" --logpath "d:\develop\mongodb\log" --replSet testrepl --keyFile "d:\develop\mongodb\relset.key" --auth

注释:

dbpath：数据存储路径  
logpath:日志存储路径

replSet:副本集名称

keyFile:key文件路径，该文件自己定义，如：abc.key，里面内容随便6-1024个长度的英文或数字

auth:表示加权限验证启动。

启动之后链接到主节点，验证之后就可以正常使用。

# 常用命令

## 登录命令:

mongo <IP:port>

## 查看所有的库

show dbs;

## 查看所有的collection

show collections;

## 查询记录

db.collectionName.find({<查询条件json>});

## 过滤字段

db.collectionName.find({<查询条件json>},{<要显示或排除的字段>});

示例:

db.User.find({\_id:ObjectId(268746238412985)},{\_id:1,name:1,age:0});

查询id为268746238412985的记录,并且只显示\_id和name字段,不显示age字段,1代表显示,0代表不显示.

# 使用mongodb3遇到的问题

## 1、单机版认证管理

### 添加用户的命令的变化:

3.0之前:db.addUser("用户名","密码");

3.0之后:db.createUser({

user:'',

pwd:'',

roles:{

role:'',

db:''

}

});

### 集成spring时开启验证的变化

3.0之前:

|  |
| --- |
| <mongo:mongo-client host="${mongo.host}" port="${mongo.port}" credentials="${mongo.credential}" id="mongo">  </mongo:mongo-client> |

3.0之后:

|  |
| --- |
| <mongo:mongo host="${mongo.host}" port="${mongo.port}" username="" password=""/> |

## 2、集群版认证管理

### 单机版和集群版开启认证的变化

单机版开启认证只需要在启动mongodb时添加--auth即可

集群版开启认证需要添加--auth并且还要给每一个副本指定keyFile,并且各个副本需要使用同一个keyFile.如何添加keyFile参考搭建副本集中的第八步.