MongoDB 单机安装：

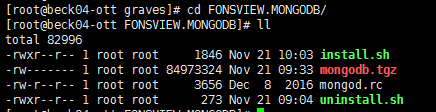
**安装部署：**

版本号：MongoDB server version: 4.0.4

1.进入目录：cd /root/install

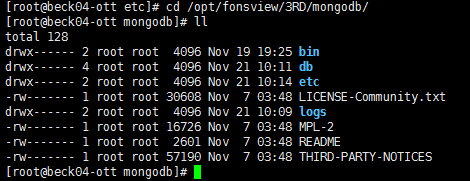
2.解压文件：tar -zxvf fonsview-mongodb-single.tgz

3.进入解压得到的文件夹：cd fonsview-mongodb-single



1. 执行安装脚本：./install.sh

默认安装位置： /opt/fonsview/3RD/mongodb



db: 存放数据

etc: 存放配置文件

logs：存放日志

1. 启动mongodb： service mongod start
2. 关闭mongodb: service mongod stop
3. 其他相关操作:

查看状态： service mongod status

重启： service mongod restart

用户密码权限设置：

版本号：MongoDB server version: 4.0.4

1. **创建root/admin用户：**

具体权限类型见附录

db.createUser({  
     user: "admin",  
     pwd: "123456",  
     roles: [{role: "userAdminAnyDatabase",db: "admin"}]  
 })

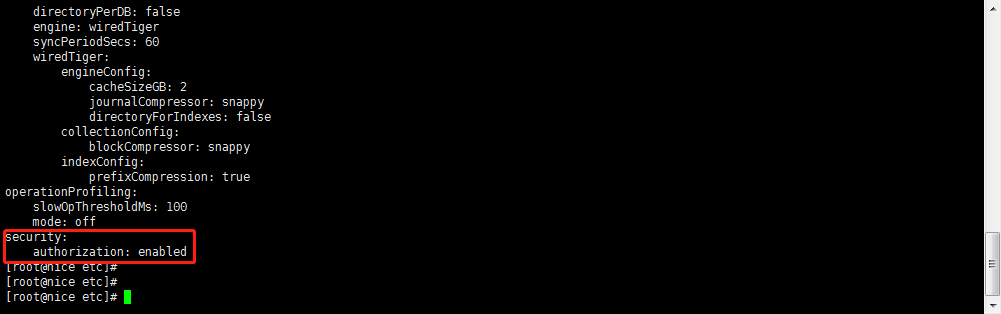
注意：userAdminAnyDatabase 权限只是针对用户管理的，对其他是没有权限的。

1. **修改mongod.conf启用授权、重启mongod服务：**

在mongod.conf 配置文件中增加如下配置

security:

authorization: enabled



修改后，重启mongod服务

1. **创建数据库读写权限用户：**

> use admin

> db.auth("admin","123456")

> use test

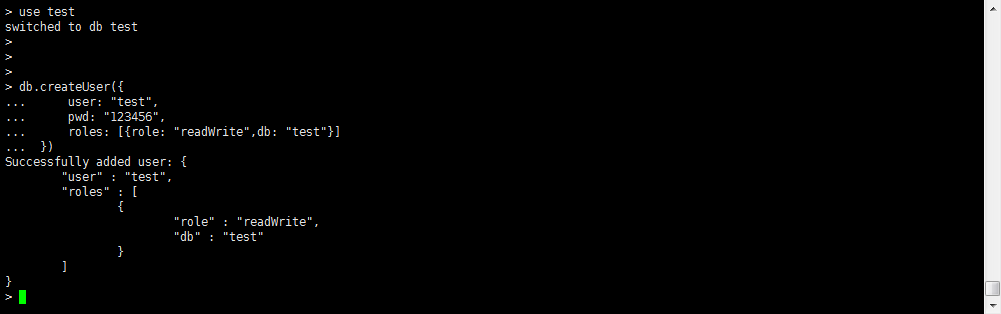
> db.createUser({

user: "test",

pwd: "123456",

roles: [{role: "readWrite",db: "test"}]

})



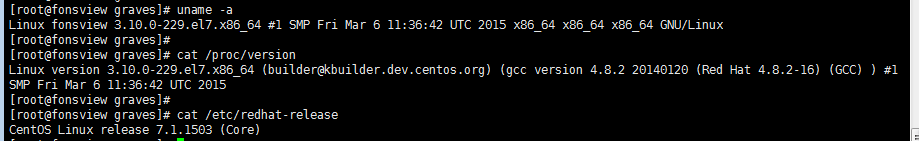
**说明：**

1. mongodb是没有默认管理员账号，所以要先添加管理员账号，再开启权限认证。  
 2. 切换到admin数据库，添加的账号才是管理员账号。  
 3. 用户只能在用户所在数据库登录，包括管理员账号。  
 4. mongo的用户是以数据库为单位来建立的，每个数据库有自己的管理员。  
 5. 管理员可以管理所有数据库，但是不能直接管理其他数据库，要先在admin数据库认证后才可以。  
 注：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

MongoDB 集群安装：

1. 环境准备：

系统：



服务器：3台（172.16.0.36,172.16.0.40,172.16.0.41）

安装包：fonsview-mongodb-cluster.tgz

|  |  |  |
| --- | --- | --- |
| 服务器A（172.16.0.36） | 服务器B（172.16.0.40） | 服务器C（172.16.0.41） |
| mongos | mongos | mongos |
| config server | config server | config server |
| shard server1 主节点 | shard server1 副节点 | shard server1 仲裁 |
| shard server2 仲裁 | shard server2 主节点 | shard server2 副节点 |
| shard server3 副节点 | shard server3 仲裁 | shard server3 主节点 |

端口分配:

mongos:23000 （DB连接端口） config:24000 shard1:25001 shard2:25002 shard3:25003

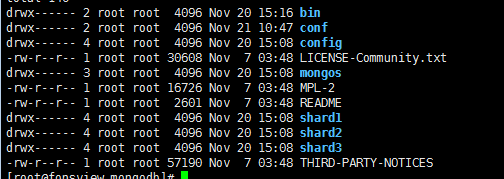
1. 安装mongodb：

进入默认目录: cd /opt/fonsview/3RD

将安装包放在默认目录下解压: tar -zxvf fonsview-mongodb-cluster.tgz

1. 分别在每台机器建立conf、mongos、config、shard1、shard2、shard3六个目录，因为mongos不存储数据，只需要建立日志文件目录即可

mkdir -p /opt/fonsview/3RD/mongodb/conf  
mkdir -p /opt/fonsview/3RD/mongodb/mongos/log  
mkdir -p /opt/fonsview/3RD/mongodb/config/data  
mkdir -p /opt/fonsview/3RD/mongodb/config/log  
mkdir -p /opt/fonsview/3RD/mongodb/shard1/data  
mkdir -p /opt/fonsview/3RD/mongodb/shard1/log  
mkdir -p /opt/fonsview/3RD/mongodb/shard2/data  
mkdir -p /opt/fonsview/3RD/mongodb/shard2/log  
mkdir -p /opt/fonsview/3RD/mongodb/shard3/data  
mkdir -p /opt/fonsview/3RD/mongodb/shard3/log



conf：存放所有服务器的配置文件

config：存放配置服务器的数据和日志信息

mongos：存放路由服务器的日志信息

shard1/shard2/shard3：存放各个分片服务器上的数据和日志信息

1. 配置服务器搭建副本集：

Mongodb3.4以后要求配置服务器也创建副本集，不然集群搭建不成功。

1. 添加配置文件:

vim /opt/fonsview/3RD/mongodb/conf/config.conf

1. 进入配置文件后添加如下信息：

## 配置文件内容  
pidfilepath=/opt/fonsview/3RD/mongodb/config/log/configsrv.pid  
dbpath=/opt/fonsview/3RD/mongodb/config/data  
logpath=/opt/fonsview/3RD/mongodb/config/log/configsrv.log  
logappend=true  
bind\_ip=0.0.0.0  
port=24000  
fork=true  
#declare this is a config db of a cluster;  
configsvr=true  
#副本集名称  
replSet=configs  
#设置最大连接数  
maxConns=2000

# 是否开启慢查询 0代表关闭，1代表只记录slowlog，2代表记录所有操作

profile=1

# 查询大于指定时间的命令

slowms=1000

#keyFile=/opt/fonsview/3RD/mongodb/conf/testKeyFile.file

#auth=true

#设置内存大小

wiredTigerCacheSizeGB=2

1. 分别启动三台服务器的config server，进入 /opt/fonsview/3RD/mongdb/bin目录下

./mongod -f /opt/fonsview/3RD/mongodb/conf/config.conf

1. 登录任意一台服务器A，初始化配置副本集, 进入/opt/fonsview/3RD/mongdb/bin 目录下

./mongo --port 24000

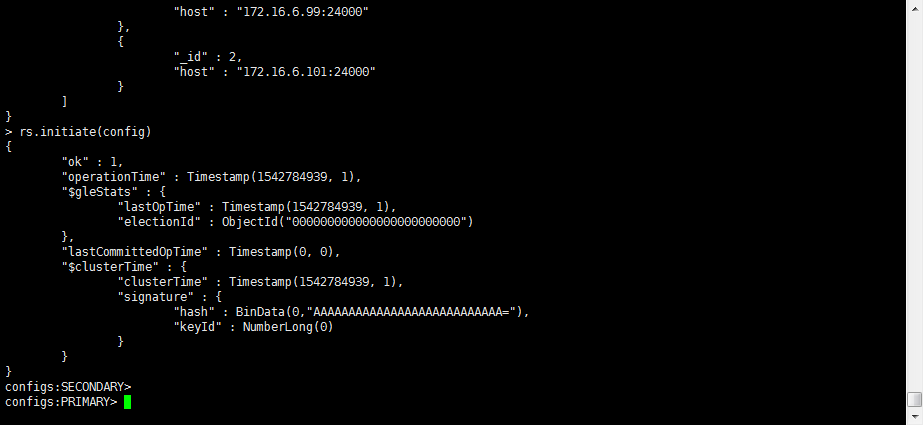
使用admin数据库,配置config变量，并初始化:

use admin;

config = {  
    \_id : "configs",  
     members : [  
         {\_id : 0, host : "172.16.0.36:24000" },  
         {\_id : 1, host : "172.16.0.40:24000" },  
         {\_id : 2, host : "172.16.0.41:24000" }  
     ]  
 };

rs.initiate(config)

初始化成功（此时 > 前已经出现 PRIMARY（主节点））：



其中的 \_id:”configs” 应与配置文件中的 replSet=congfigs 一致，host 为三个节点的 ip 和 port

如果初始化失败，会导致路由服务器和配置服务器连接不上。

1. 分片服务器搭建副本集：

配置分片副本集（三台服务器）

**设置第一个副本集：**

1. 添加配置文件:

vim /opt/fonsview/3RD/mongodb/conf/shard1.conf

1. 进入配置文件后添加如下信息：

## 配置文件内容  
pidfilepath=/opt/fonsview/3RD/mongodb/shard1/log/shard1.pid  
dbpath=/opt/fonsview/3RD/mongodb/shard1/data  
logpath=/opt/fonsview/3RD/mongodb/shard1/log/shard1.log  
logappend=true  
bind\_ip=0.0.0.0  
port=25001  
fork=true  
#副本集名称  
replSet=shard1

#声明集群分片数据库

shardsvr = true  
#设置最大连接数  
maxConns=2000

# 是否开启慢查询 0代表关闭，1代表只记录slowlog，2代表记录所有操作

profile=1

# 查询大于指定时间的命令

slowms=1000

#keyFile=/opt/fonsview/3RD/mongodb/conf/testKeyFile.file

#auth=true

#设置内存大小

wiredTigerCacheSizeGB=8

1. 分别启动三台服务器的shard1 server，进入 /opt/fonsview/3RD/mongdb/bin目录下

./mongod -f /opt/fonsview/3RD/mongodb/conf/shard1.conf

1. 登录任意一台服务器A，初始化配置副本集, 进入/opt/fonsview/3RD/mongdb/bin 目录下

./mongo --port 25001

使用admin数据库,定义副本配置，并初始化:

use admin;

config = {  
    \_id : "shard1",  
     members : [  
         {\_id : 0, host : "172.16.0.36:25001" },  
         {\_id : 1, host : "172.16.0.40:25001" },  
         {\_id : 2, host : "172.16.0.41:25001", arbiterOnly: true }  
     ]  
 };

rs.initiate(config)

第三个节点的 arbiterOnly: true 表示该节点为仲裁节点

**设置第二个副本集**：

1. 添加配置文件:

vim /opt/fonsview/3RD/mongodb/conf/shard2.conf

1. 进入配置文件后添加如下信息：

## 配置文件内容  
pidfilepath=/opt/fonsview/3RD/mongodb/shard2/log/shard2.pid  
dbpath=/opt/fonsview/3RD/mongodb/shard2/data  
logpath=/opt/fonsview/3RD/mongodb/shard2/log/shard2.log  
logappend=true  
bind\_ip=0.0.0.0  
port=25002  
fork=true

#副本集名称  
replSet=shard2

#声明集群分片数据库

shardsvr=true  
#设置最大连接数  
maxConns=2000

# 是否开启慢查询 0代表关闭，1代表只记录slowlog，2代表记录所有操作

profile=1

# 查询大于指定时间的命令

slowms=1000

#keyFile=/opt/fonsview/3RD/mongodb/conf/testKeyFile.file

#auth=true

#设置内存大小

wiredTigerCacheSizeGB=8

1. 分别启动三台服务器的shard2 server，进入 /opt/fonsview/3RD/mongdb/bin目录下

./mongod -f /opt/fonsview/3RD/mongodb/conf/shard2.conf

1. 登录任意一台服务器B，初始化配置副本集, 进入/opt/fonsview/3RD/mongdb/bin 目录下

./mongo --port 25002

使用admin数据库,定义副本配置，并初始化:

use admin;

config = {  
    \_id : "shard2",  
     members : [  
         {\_id : 0, host : "172.16.0.36:25002", arbiterOnly: true  },  
         {\_id : 1, host : "172.16.0.40:25002" },  
         {\_id : 2, host : "172.16.0.41:25002" }  
     ]  
 };

rs.initiate(config)

**设置第三个副本集：**

1. 添加配置文件:

vim /opt/fonsview/3RD/mongodb/conf/shard3.conf

1. 进入配置文件后添加如下信息：

## 配置文件内容  
pidfilepath=/opt/fonsview/3RD/mongodb/shard3/log/shard3.pid  
dbpath=/opt/fonsview/3RD/mongodb/shard3/data  
logpath=/opt/fonsview/3RD/mongodb/shard3/log/shard3.log  
logappend=true  
bind\_ip=0.0.0.0  
port=25003  
fork=true

#副本集名称  
replSet=shard3

#声明集群分片数据库

shardsvr=true  
#设置最大连接数  
maxConns=2000

# 是否开启慢查询 0代表关闭，1代表只记录slowlog，2代表记录所有操作

profile=1

# 查询大于指定时间的命令

slowms=1000

#keyFile=/opt/fonsview/3RD/mongodb/conf/testKeyFile.file

#auth=true

#设置内存大小

wiredTigerCacheSizeGB=8

1. 分别启动三台服务器的shard3 server，进入 /opt/fonsview/3RD/mongdb/bin目录下

./mongod -f /opt/fonsview/3RD/mongodb/conf/shard3.conf

1. 登录任意一台服务器C，初始化配置副本集, 进入/opt/fonsview/3RD/mongdb/bin 目录下

./mongo --port 25003

使用admin数据库,定义副本配置，并初始化:

use admin;

config = {  
    \_id : "shard3",  
     members : [  
         {\_id : 0, host : "172.16.0.36:25003" },  
         {\_id : 1, host : "172.16.0.40:25003", arbiterOnly: true },  
         {\_id : 2, host : "172.16.0.41:25003"}  
     ]  
 };

rs.initiate(config)

1. 配置路由服务器:
2. 添加配置文件:

vim /opt/fonsview/3RD/mongodb/conf/mongos.conf

1. 进入配置文件后添加如下信息：

#内容  
pidfilepath=/opt/fonsview/3RD/mongodb/mongos/log/mongos.pid  
logpath=/opt/fonsview/3RD/mongodb/mongos/log/mongos.log  
logappend=true  
bind\_ip=0.0.0.0  
port=23000  
fork=true  
#监听的配置服务器,只能有1个或者3个 configs为配置服务器的副本集名字  
configdb=configs/172.16.0.36:24000,172.16.0.40:24000,172.16.0.41:24000  
#设置最大连接数  
maxConns=2000

#keyFile=/opt/fonsview/3RD/mongodb/conf/testKeyFile.file

1. 分别启动三台服务器的mongos server，进入 /opt/fonsview/3RD/mongdb/bin目录下

./mongos -f /opt/fonsview/3RD/mongodb/conf/mongos.conf

1. 分片：

（1）mongos分片配置：

目前搭建了mongodb配置服务器、路由服务器，各个分片服务器，不过应用程序连接到mongos路由服务器并不能使用分片机制，还需要在程序里设置分片配置，让分片生效。

登录任意一台mongos，进入/opt/fonsview/3RD/mongdb/bin 目录下

./mongo --port 23000

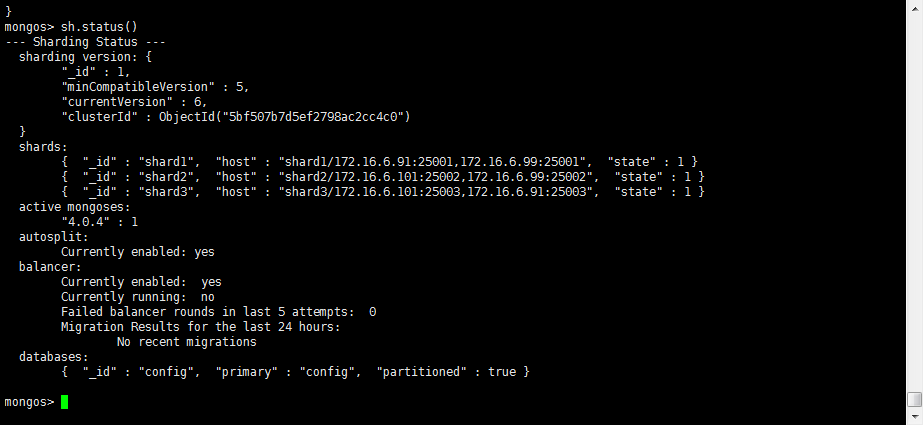
使用admin数据库，串联路由服务器与分配副本集：

use admin

sh.addShard("shard1/172.16.0.36:25001,172.16.0.40:25001,172.16.0.41:25001")  
sh.addShard("shard2/172.16.0.36:25002,172.16.0.40:25002,172.16.0.41:25002")  
sh.addShard("shard3/172.16.0.36:25003,172.16.0.40:25003,172.16.0.41:25003")

查看集群状态：

sh.status()



（2）指定需要分片的数据库：

db.runCommand( { enablesharding :"favorite"});

（3）指定数据库里需要分片的集合和片键：

db.runCommand( { shardcollection : "favorite.bookmark\_collection",key : {userId: "hashed" } } )

1. 将命令添加到系统路径中：

将命令添加到系统路径中，方便随处执行mongo命令：

1、在/etc/profile文件中，添加 export ：PATH=$PATH:/opt/fonsview/3RD/mongodb/bin

2、执行source /etc/profile，使系统环境变量立即生效

3、将mongo路径软链接到/usr/bin下：

ln -s /opt/fonsview/3RD/mongodb/bin/mongo  /usr/bin/mongo

1. 启动和关闭：

**安装部署过程中，已经执行了下列启动命令，故无需再次执行**

Mongodb的启动顺序是，先启动配置服务器，再启动分片，最后启动mongos.

./mongod -f /opt/fonsview/3RD/mongodb/conf/config.conf

./mongod -f /opt/fonsview/3RD/mongodb/conf/shard1.conf

./mongod -f /opt/fonsview/3RD/mongodb/conf/shard2.conf

./mongod -f /opt/fonsview/3RD/mongodb/conf/shard3.conf

./mongos -f /opt/fonsview/3RD/mongodb/conf/mongos.conf

关闭时，执行在进入mongos、分片和配置服务器分别执行如下命令：

use admin

db.shutdowmServer()

强烈不建议使用 kill -9 pid 方式关闭，这种方法容易造成数据丢失及服 务器瘫痪

1. 设置密码：
2. **创建用户：**

创建root用户：

use admin

db.createUser({

     user: "admin",

     pwd: "123456",

     roles: [{role: "userAdminAnyDatabase",db: "admin"}]

})

创建指定数据库用户：

use favorite

db.createUser({

     user: "favorite",

     pwd: "123456",

     roles: [{role: "readWrite",db: "favorite"}]

})

1. **生成密钥文件：**

openssl rand -base64 756 > /data/mongodb/testKeyFile.file

chmod 400 /data/mongodb/keyfile/testKeyFile.file

第一条命令是生成密钥文件，第二条命令是使用chmod更改文件权限，为文件所有者提供读权限

1. **开启验证**

将配置文件中以下两行配置注释放开

#keyFile=/opt/fonsview/3RD/mongodb/conf/testKeyFile.file

#auth=true

这个步骤比较重要。设置访问控制有两种方式。我选择在配置文件里面配置好。（也可以在启动命令时使用命令来指定）

   依次在每台机器上的mongod（注意是所有的mongod不是mongos）的配置文件中加入下面一段配置。如我在10.12.40.83上的config server，shard1，shard2，shard3都加入下面的配置文件

security:

  keyFile: /data/mongodb/testKeyFile.file

  authorization: enabled

  依次在每台机器上的mongos配置文件中加入下面一段配置。如我在10.12.40.83上的mongos配置文件中加入上面的一段配置

security:

  keyFile: /data/mongodb/testKeyFile.file

解释：

    mongos比mongod少了authorization：enabled的配置。原因是，副本集加分片的安全认证需要配置两方面的，副本集各个节点之间使用内部身份验证，用于内部各个mongo实例的通信，只有相同keyfile才能相互访问。所以都要开启keyFile: /data/mongodb/testKeyFile.file

    然而对于所有的mongod，才是真正的保存数据的分片。mongos只做路由，不保存数据。所以所有的mongod开启访问数据的授权authorization:enabled。这样用户只有账号密码正确才能访问到数据

1. **重启所有服务**

重启成功后，操作时，需要先使用 db.auth('root','123456') 命令来验证身份，返回值 0:验证失败 1：验证成功

1. 常用命令：

在数据库 admin 下的执行下列命令:

**db.runCommand( { movePrimary : "favorite", to : "shard1" } )** ：使用 movePrimary 命令变更数据库默认的Primary shard，非分片集合将会从当前shard移动到新的主分片。

**db.adminCommand({"flushRouterConfig":1})**：在使用movePrimary命令变更数据库的主分片之后，config server中的配置信息是最新的，mongos缓存的配置信息变得过时了。MongoDB提供命令：flushRouterConfig 强制mongos从config server获取最新的配置信息，刷新mongos的缓存。

**db.serverStatus().connections：**查看DB的最大连接数，current + available 即为最大连接数。如果比设置的最大连接数小：是因为Linux默认进程能打开最大文件数有关，可以通过ulimit 解决



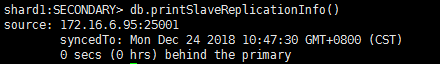
在数据库 testdb 下的执行下列命令:

**help**: 查看帮助

**rs.status()**：查看复制集状态

**rs.printReplicationInfo()**:查看oplog状态

:查看复制集延迟



* source——从库的IP及端口
* syncedTo——当前的同步情况，延迟了多久等信息

**use testdb** :切换 testdb 数据库

**show collections** : 查看当前数据库下的所有集合

**db** : 查看当前数据库名称

**db.serverStatus()**:查看服务状态详情

**db.stats()**:查询当前数据库统计信息

在数据库 testdb 中的集合 testcollection 下执行下列命令：

**db.testcollection.count()** : 获取当前集合文档总数

**db.testcollection.find().size(m).limit(n)** : 从第m条开始查询n条记录

**db.testcollection.getShardDistribution()** : 查看分片情况

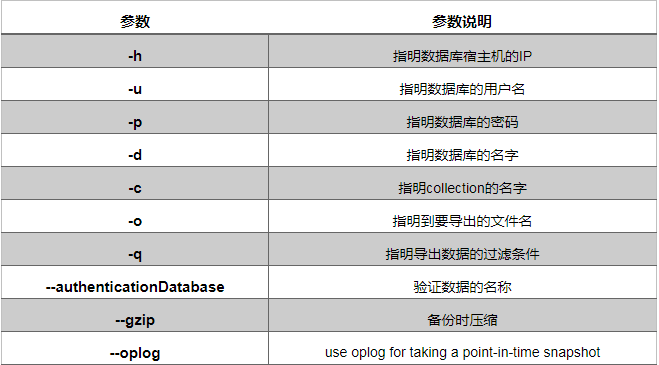
1. 数据备份与恢复：

1 mongodump/mongorestore：

1. 利用mongodump进行备份:

mongodump -h **172.16.0.36**:**23000** -u root -p root --authenticationDatabase favorite -o /opt/fonsview/3RD/dbBackup

若无密码，则 -u -p 省略

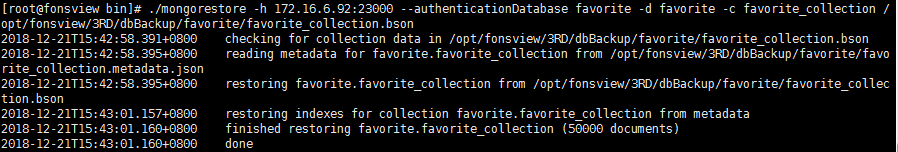
命令说明： 

1. 利用mongorestore进行恢复：

mongorestore -h **172.16.0.36**:**23000** -u root -p root --authenticationDatabase favorite -d test -c favorite\_collection /opt/fonsview/3RD/dbBackup/favorite\_collection.bson

命令说明：

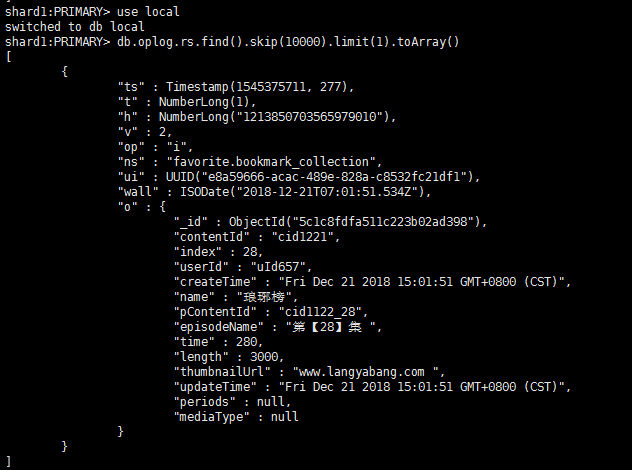
成功恢复结果如下图所示：



需要重启mongos

2 oplog:

MongoDB 的Replication是通过一个日志来存储写操作的，这个日志就叫做oplog。Secondary就是通过查看Primary 的oplog这个集合来进行复制的。每个节点都有oplog，记录这从主节点复制过来的信息，这样每个成员都可以作为同步源给其他节点。

oplog的数据结构如下：

* ts: 8字节的时间戳，由4字节unix timestamp + 4字节自增计数表示。
* op：1字节的操作类型
  + "i"： insert
  + "u"： update
  + "d"： delete
  + "c"： db cmd
  + "db"：声明当前数据库 (其中ns 被设置成为=>数据库名称+ '.')
  + "n": no op,即空操作，其会定期执行以确保时效性
* ns：操作所在的namespace
* o：操作所对应的document，即当前操作的内容（比如更新操作时要更新的的字段和值）
* o2: 在执行更新操作时的where条件，仅限于update时才有该属性
* **全量恢复：**

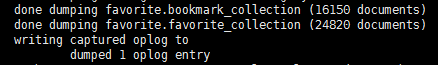
首先分别导出各分区的 oplog

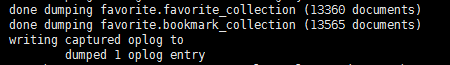
mongodump -h 127.0.0.1 --port 25001 --oplog -o /home/mongod/backup/oplog

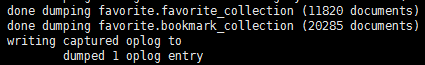
mongodump -h 127.0.0.1 --port 25002 --oplog -o /home/mongod/backup/oplog

mongodump -h 127.0.0.1 --port 25003 --oplog -o /home/mongod/backup/oplog

操作成功：

Shard1：

Shard2：

Shard3：

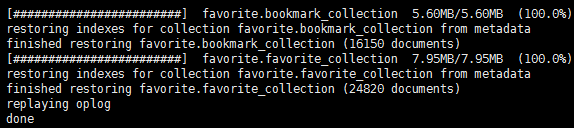
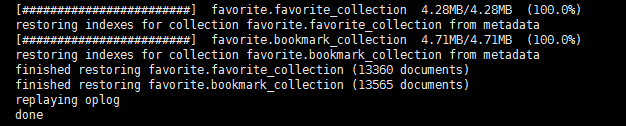
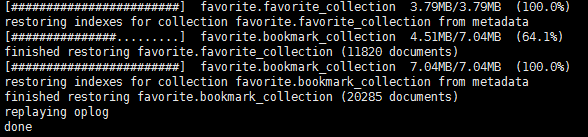
操作日志获取成功后，使用 mongorestore 命令即可恢复：

mongorestore -h 127.0.0.1 --port 25001 --oplogReplay --drop /home/mongod/backup/oplog

mongorestore -h 127.0.0.1 --port 25002 --oplogReplay --drop /home/mongod/backup/oplog

mongorestore -h 127.0.0.1 --port 25003 --oplogReplay --drop /home/mongod/backup/oplog

恢复成功截图如下：

Shard1:Shard2:Shard3:

* **即时恢复PITR（Point-In-Time-Recovery）：**

前提：

需要恢复的时间点在 oplog 时间域中

模拟错误操作删除集合：

db.bookmark\_collection.drop()

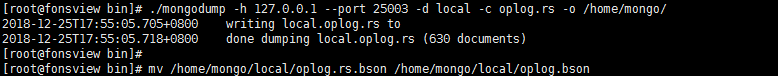
集合删除后，原有的分片信息也会被删除，因此需要重新指定数据库里需要分片的集合和片键：

db.runCommand( { shardcollection : "favorite.bookmark\_collection",key : {userId: "hashed" } } )

导出local 库下的 oplog.rs 集合并对 oplog 重命名：

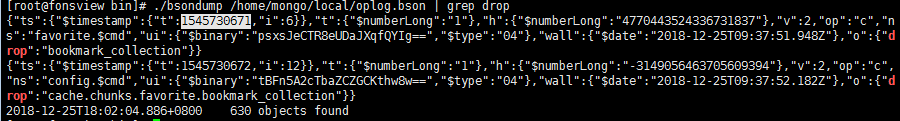
./mongodump -h 127.0.0.1 --port 25003 -d local -c oplog.rs -o /home/mongo/

mv /home/mongo/local/oplog.rs.bson /home/mongo/local/oplog.bson



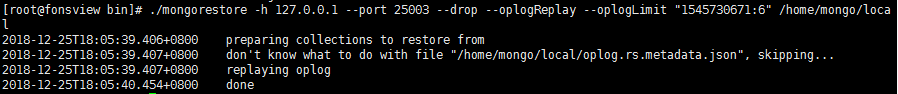
在bson文件中查找时间点：

./bsondump /home/mongo/local/oplog.bson | grep drop



执行还原操作：

./mongorestore -h 127.0.0.1 --port 25001 --oplogReplay --oplogLimit "1545787867:4" /home/mongo/local/



还原成功：



1. 常见问题：

若三台分片所在的一台发生故障（如：重启），不会影响数据库的运行。在发生故障的那台重新启动conf服务、mongod服务和mongos服务后按照如下步骤设置主节点优先级即可（如主、副节点未发生变化请忽略）。

在设置mongodb副本集时，Primary节点，second节点，仲裁节点，有可能资源配置（CPU或者内存）不均衡，所以要求某些节点不能成为Primary  
我们知道mongodb的设置：  
  除了仲裁节点，其他每个节点都有个优先权，可以手动设置优先权来决定谁的成为primay的权重最大。  
  副本集中通过设置priority的值来决定优先权的大小，这个值的范围是0--100，值越大，优先权越高。  
默认的值是1，rs.conf是不显示的；  
如果值是0，那么不能成为primay。  
1.规划时直接设置，这个就略过了  
2.在线加入的节点配置：  
配置过程：  
通过修改priority的值来实现（默认的优先级是1（0-100），priority的值设的越大，就优先成为主）  
cfg=rs.conf()  
cfg.members[0].priority = 2  
rs.reconfig(cfg)  
注意：第2步members大括号里面的成员和\_id是没有关系的，而是rs.conf查出来节点的数值的顺序；

这些操作必须在Primary上进程。

MongoDB应该分配的内存大小最好满足内存大小>索引+热数据+连接占用内存

默认每个连接数占用10M内存

****分片集群是如何处理查询的。通过分片键将查询路由给指定分片，一旦到了某个分片上，由分片自行决定使用哪个索引来执行该查询。在为应用程序设计查询和索引时，请牢记这一点。****

附录：

**MongoDB内置角色介绍：**

#### **(1).数据库用户角色**

针对每一个数据库进行控制。  
 **read**:提供了读取所有非系统集合，以及系统集合中的system.indexes, system.js, system.namespaces  
 **readWrite**: 包含了所有read权限，以及修改所有非系统集合的和系统集合中的system.js的权限.

#### **(2).数据库管理角色**

每一个数据库包含了下面的数据库管理角色。  
 **dbOwner**：该数据库的所有者，具有该数据库的全部权限。  
 **dbAdmin**：一些数据库对象的管理操作，但是没有数据库的读写权限。（参考：[http://docs.mongodb.org/manual/reference/built-in-roles/#dbAdmin](http://docs.mongodb.org/manual/reference/built-in-roles/" \l "dbAdmin" \t "https://www.cnblogs.com/zzw1787044/p/_blank)）  
 **userAdmin**：为当前用户创建、修改用户和角色。拥有userAdmin权限的用户可以将该数据库的任意权限赋予任意的用户。

#### **(3).集群管理权限**

admin数据库包含了下面的角色，用户管理整个系统，而非单个数据库。这些权限包含了复制集和共享集群的管理函数。  
 **clusterAdmin**：提供了最大的集群管理功能。相当于clusterManager, clusterMonitor, and hostManager和dropDatabase的权限组合。  
 **clusterManager**：提供了集群和复制集管理和监控操作。拥有该权限的用户可以操作config和local数据库（即分片和复制功能）  
 **clusterMonitor**：仅仅监控集群和复制集。  
 **hostManager**：提供了监控和管理服务器的权限，包括shutdown节点，logrotate, repairDatabase等。  
 备份恢复权限：admin数据库中包含了备份恢复数据的角色。包括backup、restore等等。

#### **(4).所有数据库角色**

admin数据库提供了一个mongod实例中所有数据库的权限角色：  
 **readAnyDatabase**：具有read每一个数据库权限。但是不包括应用到集群中的数据库。  
 **readWriteAnyDatabase**：具有readWrite每一个数据库权限。但是不包括应用到集群中的数据库。  
 **userAdminAnyDatabase**：具有userAdmin每一个数据库权限，但是不包括应用到集群中的数据库。  
 **dbAdminAnyDatabase**：提供了dbAdmin每一个数据库权限，但是不包括应用到集群中的数据库。

#### **(5). 超级管理员权限**

root: dbadmin到admin数据库、useradmin到admin数据库以及 UserAdminAnyDatabase。但它不具有备份恢复、直接操作system.\*集合的权限，但是拥有root权限的超级用户可以自己给自己赋予这些权限。

**用户相关命令：**

创建用户：db.createUser()

获得数据库的所有用户权限信息：db.getUsers()  
 获得某个用户的权限信息：db.getUser()  
 删除用户：db.dropUser()  
 删除所有用户:db.dropAllUsers()  
 将一个角色赋予给用户：db.grantRolesToUser()  
 撤销某个用户的某个角色权限：db.revokeRolesFromUser()  
 更改密码：db.changeUserPassword()