MongoDB操作手册

一、安装

```
#! /bin/bash
cat > /etc/yum.repos.d/mongodb-org-3.6.repo <<'eof'
[mongodb-org-3.6]
name=MongoDB Repository
baseurl=https://repo.mongodb.org/yum/redhat/$releasever/mongodb-org/3.6/x86_64/
gpgcheck=1
enabled=1
gpgkey=https://www.mongodb.org/static/pgp/server-3.6.asc
eof

sudo yum install -y mongodb-org

sudo systemctl start mongod

sudo systemctl status mongod

sudo systemctl enable mongod

mongo --host 127.0.0.1:27017
```

1.mongodb 默认存在的库

> test:登录时默认存在的库

2.管理MongoDB有关的系统库

> admin库:系统预留库,MongoDB系统管理库
> local库:本地预留库,存储关键日志
> config库:MongoDB配置信息库

二、基础操作

Mongo的查询命令

```
# 查看当前所有数据库
show dbs
# 查看集合
show tables/show collections
# 查询数据行数:
db.log.count()
# 全表查询:
db.log.find()
# 每页显示50条记录:
DBQuery.shellBatchSize=50;
# 按照条件查询
db.log.find({uid:999})
# 以标准的json格式显示数据
> db.log.find({uid:999}).pretty()
{
    "_id" : ObjectId("5cc516e60d13144c89dead33"),
    "uid" : 999,
    "name" : "mongodb",
    "age" : 6,
    "date" : ISODate("2019-04-28T02:58:46.109Z")
}
# 查看集合存储信息
app> db.log.totalSize()          //集合中索引+数据压缩存储之后的大小
```

1.创建数据库

```
# 如果数据库不存在，则创建数据库，否则切换到指定数据库。
use DATABASE_NAME

# 对于管理员用户,必须在admin下创建.
#1. 建用户时,use到的库,就是此用户的验证库
#2. 登录时,必须明确指定验证库才能登录
#3. 通常,管理员用的验证库是admin,普通用户的验证库一般是所管理的库设置为验证库
#4. 如果直接登录到数据库,不进行use,默认的验证库是test,不是我们生产建议的.
#5. 从3.6 版本开始，不添加bindIp参数，默认不让远程登录，只能本地管理员登录。
```

```
> use superwei
switched to db superwei
> db
superwei
>
```

1）使用show dbs 可以查看当前所有的数据库,使用show tables/show collections 查看数据合集

```
show dbs                          ##查看当前所有数据库
show tables/show collections      ##查看集合
```

```
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
weiwei  0.000GB
>
```

2）可以看到刚刚创建的superwei库并没有在数据库列表中，这是因为superwei库中没有存在数据，如果我们想要superwei库存在于数据库列表中，就需要向其中插入一些数据，例如：

```
db.create.insert({"name":"伍嘉威"})
```

```
> db.create.insert({"name":"伍嘉威"})
WriteResult({ "nInserted" : 1 })
>
```

3）这时候就可以在数据库列表显示superwei库了。

```
> show dbs
admin     0.000GB
config    0.000GB
local     0.000GB
superwei  0.000GB
weiwei    0.000GB
>
```

2.创建集合

1）MongoDB创建集合的语法如下：

```
db.createCollection(name, options)
参数说明：
     name：要创建的集合名称
     options：可选参数，指定内存大小及索引的选项
option 的参数选择：
        max （数值） （可选）指定固定集合中包含文档的最大数量。
        size （数值） （可选）为固定集合指定一个最大值，即字节数。如果 capped 为 true，也需要指定该字段。
        capped （布尔型）如果为 true，则创建固定集合。固定集合是指有着固定大小的集合，当达到最大值时，它会自动覆盖最早的文档。当该值为 true 时，必须指定 size 参数。
        autoIndexId （布尔型）3.2 之后不再支持该参数。如为 true，自动在 _id 字段创建索引。默认为 false。
```

例如：

```
> db.createCollection( "嘉威" )
{ "ok" : 1 }
> show collections
嘉威
>
```

2）MongoDB创建集合带参数的写法如下：

创建固定集合ccc，整个集合空间大小 6142800 B, 文档最大个数为 10000 个。

```
db.createCollection("ccc", { capped : true, autoIndexId : true, size : 6142800, max : 10000 } )
```

```
> db.createCollection("ccc", { capped : true, autoIndexId : true, size : 6142800, max : 10000 } )
{
        "note" : "the autoIndexId option is deprecated and will be removed in a future release",
        "ok" : 1
}
>
```

3）当你插入一些文档时，MongoDB 会自动创建集合

```
db.www.insert({"name" : "菜鸟"})
```

```
> db.www.insert({"name" : "菜鸟"})
WriteResult({ "nInserted" : 1 })
> show collections
ccc
www
嘉威
嘉威威
>
```

3.删除数据库

1）MongoDB删除语法如下所示：

```
# 删除当前数据库，可以用db查看当前数据库名。
db.dropDatabase()

# 例如我们要删除weiwei这个库。
>use weiwei
>db.dropDatabase()
{ "dropped" : "weiwei", "ok" : 1 }
```

```
> show dbs
admin     0.000GB
config    0.000GB
local     0.000GB
superwei  0.000GB
weiwei    0.000GB
> use weiwei
switched to db weiwei
> db.dropDatabase()
{ "dropped" : "weiwei", "ok" : 1 }
> show dbs
admin     0.000GB
config    0.000GB
local     0.000GB
superwei  0.000GB
>
```

2）MongoDB删除集群的语法格式：

```
use ccc
db.collenction.drop()
```

```
> show collections
ccc
www
嘉威
嘉威威
> db.www.drop()
true
> show collections
ccc
嘉威
嘉威威
>
```

4.插入文档

1）MongoDB 使用 insert() 或 save() 方法向集合中插入文档，语法如下：

```
db."COLLECTION_NAME".insert(document)
或
db."COLLECTION_NAME".save(document)

- insert():      若插入的数据主键已经存在，则会抛**org.springframework.dao.DuplicateKeyException** 异常，提示主键重复，不保存当前数据。
- save():        如果 _id 主键存在则更新数据，如果不存在就插入数据。该方法新版本中已废弃，可以使用 db.collection.insertOne() 或 db.collection.replaceOne() 来代替。
```

2）3.2 版本之后新增了 db.collection.insertOne() 和 db.collection.insertMany()。

```
db."collection".insertOne() 用于向集合插入一个新文档，语法格式如下：
db."collection".insertOne(
   <document>,
   {
      writeConcern: <document>
   }
)

db."collection".insertMany() 用于向集合插入一个多个文档，语法格式如下：
db."collection".insertMany(
   [ <document 1> , <document 2>, ... ],
   {
      writeConcern: <document>,
      ordered: <boolean>
   }
)

# 参数说明：
# document：要写入的文档。
```

```
# writeConcern：写入策略，默认为 1，即要求确认写操作，0 是不要求。
# ordered：指定是否按顺序写入，默认 true，按顺序写入。
```

3）查看已插入文档

```
db."collection".find()
```

5.更新文档

1）update() 方法用于更新已经存在的文档。语法格式如下：

```
db."collection".update(
    <query>,
    <update>,
    {
      upsert: <boolean>,
      multi: <boolean>,
      writeConcern: <document>
    }
)

# 参数说明：
query : # update的查询条件，类似sql update查询内where后面的。
update : # update的对象和一些更新的操作符（如$,$inc...）等，也可以理解为sql update查询内set后面的
upsert : # 可选，这个参数的意思是，如果不存在update的记录，是否插入objNew,true为插入，默认是false，不插入。
multi : # 可选，默认是false,只更新找到的第一条记录，如果这个参数为true,就把按条件查出来多条记录全部更新。
writeConcern : # 可选，抛出异常的级别。
```

2）save() 方法通过传入的文档来替换已有文档，_id 主键存在就更新，不存在就插入。语法格式如下：

```
db."collection".save(
    <document>,
    {
      writeConcern: <document>
    }
)

# 参数说明:
document : # 文档数据。
writeConcern : # 可选，抛出异常的级别。
```

6.删除文档

1）remove() 方法的基本语法格式如下所示:

```
2.6版本之后:
db."collection".remove(
    <query>,
    {
      justOne: <boolean>,
      writeConcern: <document>
    }
)
2.6版本之前:
db."collection".remove(
    <query>,
    <justOne>
)

#参数说明:
query : #  （可选）删除的文档的条件。
justOne : #（可选）设为true或1，则只删除一个文档，不设置该参数或使用默认值false，则删除所有匹配条件的文档。
writeConcern : #（可选）抛出异常的级别。
```

2）如果你只想删除第一条找到的记录可以设置 justOne 为 1，如下所示:

```
db."COLLECTION_NAME".remove(DELETION_CRITERIA,1)
```

3）如果你想删除所有数据，可以使用以下方式（类似常规 SQL 的 truncate 命令）：

```
db."COLLECTION_NAME".remove({})
```

7.查询文档

1）MongoDB 查询数据的语法格式如下：

```
db."collection".find(query, projection)

query ：# 可选，使用查询操作符指定查询条件
projection ：# 可选，使用投影操作符指定返回的键。查询时返回文档中所有键值， 只需省略该参数即可（默认省略）
```

2）如果你需要以易读的方式来读取数据，可以使用 pretty() 方法，语法格式如下：

```
db."collection".find().pretty()

# 除了 find() 方法之外，还有一个 findOne() 方法，它只返回一个文档。
```

3）AND 条件：MongoDB 的 find() 方法可以传入多个键(key)，每个键(key)以逗号隔开，即常规 SQL 的 AND 条件。语法：

```
db."collection".find({key1:value1, key2:value2}).pretty()
```

4）OR 条件语句使用了关键字 $or,语法格式如下：

```
db."collection".find(
   {
      $or: [
         {key1: value1}, {key2:value2}
      ]
   }
).pretty()
```

AND条件语句还可以与OR条件语句结合起来使用。

8.条件操作符

MongoDB中条件操作符有:

```
    (>) 大于 - $gt
    (<) 小于 - $lt
    (>=) 大于等于 - $gte
    (<= ) 小于等于 - $lte
```

1）如果你想获取 "col" 集合中 "likes" 大于 100 的数据，你可以使用以下命令:

```
db.col.find({likes : {$gt : 100}})          //类似于
Select * from col where likes > 100;
```

2）如果你想获取"col"集合中 "likes" 大于等于 100 的数据，你可以使用以下命令:

```
db.col.find({likes : {$gte : 100}})          //类似于
Select * from col where likes >=100;
```

3）如果你想获取"col"集合中 "likes" 小于 150 的数据，你可以使用以下命令:

```
db.col.find({likes : {$lt : 150}})          //类似于
Select * from col where likes < 150;
```

4）如果你想获取"col"集合中 "likes" 小于等于 150 的数据，你可以使用以下命令:

```
db.col.find({likes : {$lte : 150}})          //类似于
Select * from col where likes <= 150;
```

5）如果你想获取"col"集合中 "likes" 大于100，小于 200 的数据，你可以使用以下命令：

```
db.col.find({likes : {$lt :200, $gt : 100}})  //类似于
Select * from col where likes>100 AND  likes<200;
```

二、MongoDB补充知识

1.mongodb的关闭方式

```
/bin/mongod -f  /mongodb/38025/conf/mongodb.conf --
shutdown
```

2.mongodb对象操作

```
mongo          mysql
库     ----->  库

集合   ----->  表

文档   ----->  数据行
```

3.

三、MongoDB复制集RS（ReplicationSet）

1.基本原理：

> 基本构成是1主2从的结构，自带互相监控投票机制（Raft（MongoDB） Paxos（mysql MGR 用的是变种））
> 如果发生主库宕机，复制集内部会进行投票选举，选择一个新的主库替代原有主库对外提供服务。同时复制集会自动通知
> 客户端程序，主库已经发生切换了。应用就会连接到新的主库。

2.Replication Set配置过程详解

1）规划

> 三个以上的mongodb节点（或多实例）

2）环境准备

> 多个端口：
> 28017、28018、28019、28020

3）多套目录：

```
mkdir -p /mongodb/28017/conf /mongodb/28017/data /mongodb/28017/log
mkdir -p /mongodb/28018/conf /mongodb/28018/data /mongodb/28018/log
mkdir -p /mongodb/28019/conf /mongodb/28019/data /mongodb/28019/log
mkdir -p /mongodb/28020/conf /mongodb/28020/data /mongodb/28020/log
```

4）多套配置文件：

```
/mongodb/28017/conf/mongod.conf
/mongodb/28018/conf/mongod.conf
/mongodb/28019/conf/mongod.conf
/mongodb/28020/conf/mongod.conf
```

5）配置文件内容：

```
cat > /mongodb/28017/conf/mongod.conf <<EOF
systemLog:
  destination: file
  path: /mongodb/28017/log/mongodb.log
  logAppend: true
storage:
  journal:
    enabled: true
  dbPath: /mongodb/28017/data
  directoryPerDB: true
```

```
  \#engine: wiredTiger
  wiredTiger:
    engineConfig:
      cacheSizeGB: 1
      directoryForIndexes: true
    collectionConfig:
      blockCompressor: zlib
    indexConfig:
      prefixCompression: true
processManagement:
  fork: true
net:
  bindIp: 127.0.0.1
  port: 28017
replication:
  oplogSizeMB: 2048
  replSetName: my_repl
EOF

\cp  /mongodb/28017/conf/mongod.conf
/mongodb/28018/conf/
\cp  /mongodb/28017/conf/mongod.conf
/mongodb/28019/conf/
\cp  /mongodb/28017/conf/mongod.conf
/mongodb/28020/conf/

sed 's#28017#28018#g' /mongodb/28018/conf/mongod.conf -
i
sed 's#28017#28019#g' /mongodb/28019/conf/mongod.conf -
i
sed 's#28017#28020#g' /mongodb/28020/conf/mongod.conf -
i
chown -R mongod:mongod /mongodb
```

6）启动多个实例备用：

```
/bin/mongod -f /mongodb/28017/conf/mongod.conf
/bin/mongod -f /mongodb/28018/conf/mongod.conf
/bin/mongod -f /mongodb/28019/conf/mongod.conf
/bin/mongod  -f /mongodb/28020/conf/mongod.conf

netstat -lnp|grep 280
```

7）配置普通复制集：

```
# 1主2从，从库普通从库
mongo --port 28017 admin
config = {_id: 'my_repl', members: [
                           {_id: 0, host:
'127.0.0.1:28017'},
                           {_id: 1, host:
'127.0.0.1:28018'},
                           {_id: 2, host:
'127.0.0.1:28019'}]
          }
rs.initiate(config)
# 查询复制集状态：
rs.status();

或

# 1主1从1个arbiter
mongo -port 28017 admin
config = {_id: 'my_repl', members: [
                           {_id: 0, host:
'127.0.0.1:28017'},
                           {_id: 1, host:
'127.0.0.1:28018'},
                           {_id: 2, host:
'127.0.0.1:28019',"arbiterOnly":true}]
          }
rs.initiate(config)
```

3.复制集管理操作

1）查看复制集状态

```
rs.status();     //查看整体复制集状态
rs.isMaster(); // 查看当前是否是主节点
rs.conf();    //查看复制集配置信息
```

2）添加删除节点

```
rs.remove("ip:port"); // 删除一个节点
rs.add("ip:port"); // 新增从节点
rs.addArb("ip:port"); // 新增仲裁节点
```

例子：添加 arbiter节点

1、连接到主节点

```
mongo --port 28018 admin
```

2、添加仲裁节点

```
my_repl:PRIMARY> rs.addArb("192.168.75.33:28020")
```

3、查看节点状态

```
my_repl:PRIMARY> rs.isMaster
({
    "hosts" : [
        "192.168.75.33:28017",
        "192.168.75.33:28018",
        "192.168.75.33:28019"
    ],
    "arbiters" : [
        "192.168.75.33:28020"
    ]
})

rs.remove("ip:port"); // 删除一个节点
例子：
```

```
my_repl:PRIMARY> rs.remove("192.168.75.33:28019");
{ "ok" : 1 }
my_repl:PRIMARY> rs.isMaster()
rs.add("ip:port"); // 新增从节点
例子：
my_repl:PRIMARY> rs.add("192.168.75.33:28019")
{ "ok" : 1 }
```

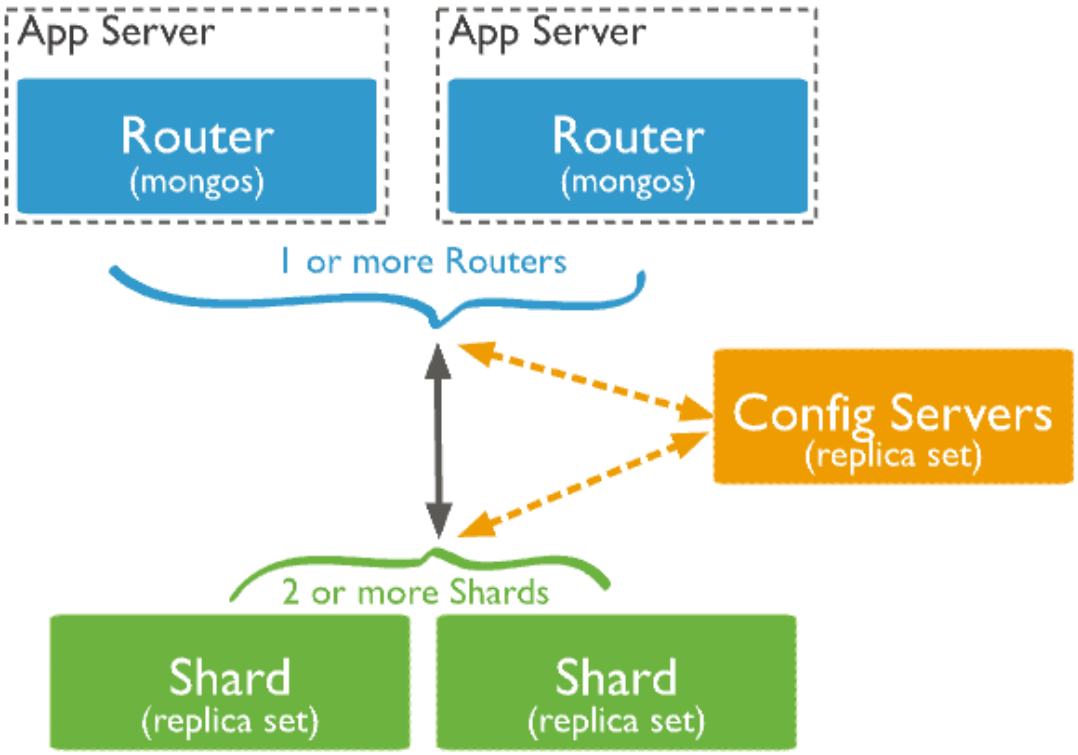
4.特殊从节点

介绍：

> arbiter节点：主要负责选主过程中的投票，但是不存储任何数据，也不提供任何服务
> hidden节点：隐藏节点，不参与选主，也不对外提供服务。
> delay节点：延时节点，数据落后于主库一段时间，因为数据是延时的，也不应该提供服务或参与选主，所以通常会配合hidden（隐藏）
>
> 一般情况下会将delay+hidden一起配置使用

四、MongoDB Sharding Cluster 分片集群搭建

下图描述了分片集群中组件的交互：



1.规划

> 10个实例：38017-38026

(1) configserver:38018-38020

> 3台构成的复制集（1主两从，不支持arbiter）38018-38020（复制集名字configsvr）

(2) shard节点:

> sh1：38021-23　（1主两从，其中一个节点为arbiter，复制集名字sh1）
> sh2：38024-26　（1主两从，其中一个节点为arbiter，复制集名字sh2）

(3) mongos:

> 38017

2.Shard节点配置过程

1）目录创建：

```
mkdir -p /mongodb/38021/conf  /mongodb/38021/log /mongodb/38021/data
mkdir -p /mongodb/38022/conf  /mongodb/38022/log /mongodb/38022/data
mkdir -p /mongodb/38023/conf  /mongodb/38023/log /mongodb/38023/data
mkdir -p /mongodb/38024/conf  /mongodb/38024/log /mongodb/38024/data
mkdir -p /mongodb/38025/conf  /mongodb/38025/log /mongodb/38025/data
mkdir -p /mongodb/38026/conf  /mongodb/38026/log /mongodb/38026/data
```

2）修改配置文件：

第一组复制集搭建：21-23（1主 1从 1Arb）

```
cat >  /mongodb/38021/conf/mongodb.conf  <<EOF
systemLog:
  destination: file
  path: /mongodb/38021/log/mongodb.log
  logAppend: true
storage:
  journal:
    enabled: true
  dbPath: /mongodb/38021/data
  directoryPerDB: true
  \#engine: wiredTiger
  wiredTiger:
    engineConfig:
      cacheSizeGB: 1
      directoryForIndexes: true
    collectionConfig:
      blockCompressor: zlib
    indexConfig:
      prefixCompression: true
```

```
net:
  bindIp: 127.0.0.1
  port: 38021
replication:
  oplogSizeMB: 2048
  replSetName: sh1
sharding:
  clusterRole: shardsvr
processManagement:
  fork: true
EOF

\cp  /mongodb/38021/conf/mongodb.conf  /mongodb/38022/conf/
\cp  /mongodb/38021/conf/mongodb.conf  /mongodb/38023/conf/

sed 's#38021#38022#g' /mongodb/38022/conf/mongodb.conf -i
sed 's#38021#38023#g' /mongodb/38023/conf/mongodb.conf -i
```

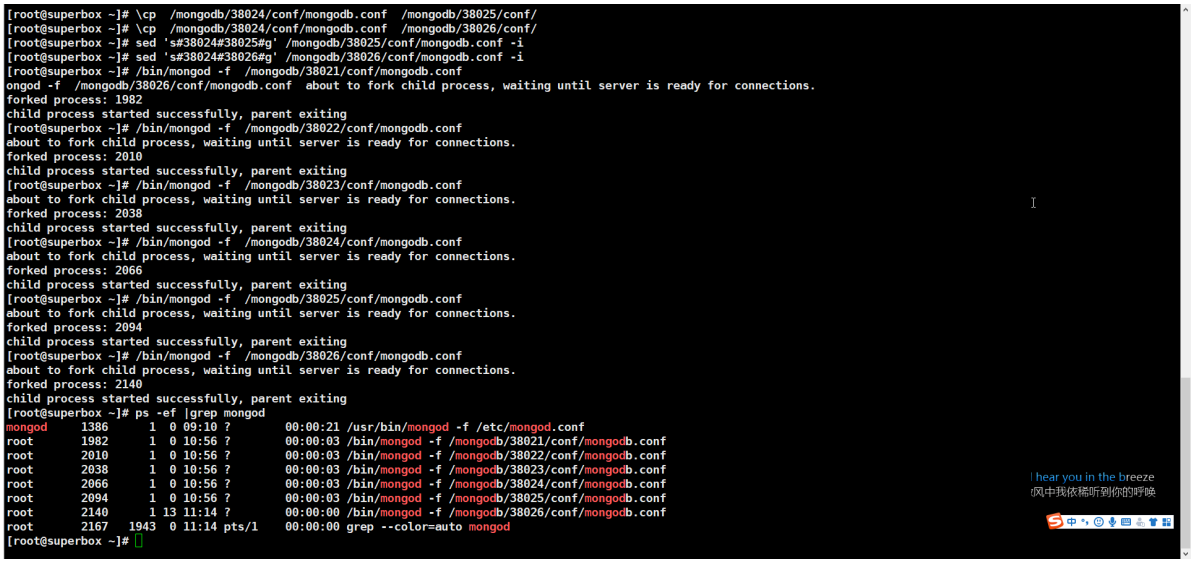
第二组节点：24-26(1主1从1Arb)

```
cat > /mongodb/38024/conf/mongodb.conf <<EOF
systemLog:
  destination: file
  path: /mongodb/38024/log/mongodb.log
  logAppend: true
storage:
  journal:
    enabled: true
  dbPath: /mongodb/38024/data
  directoryPerDB: true
  wiredTiger:
    engineConfig:
      cacheSizeGB: 1
      directoryForIndexes: true
    collectionConfig:
      blockCompressor: zlib
```

```
    indexConfig:
      prefixCompression: true
net:
  bindIp: 127.0.0.1
  port: 38024
replication:
  oplogSizeMB: 2048
  replSetName: sh2
sharding:
  clusterRole: shardsvr
processManagement:
  fork: true
EOF

\cp  /mongodb/38024/conf/mongodb.conf /mongodb/38025/conf/
\cp  /mongodb/38024/conf/mongodb.conf /mongodb/38026/conf/

sed 's#38024#38025#g' /mongodb/38025/conf/mongodb.conf -i
sed 's#38024#38026#g' /mongodb/38026/conf/mongodb.conf -i
```

3）启动所有节点，并搭建复制集

```
/bin/mongod -f  /mongodb/38021/conf/mongodb.conf
/bin/mongod -f  /mongodb/38022/conf/mongodb.conf
/bin/mongod -f  /mongodb/38023/conf/mongodb.conf
/bin/mongod -f  /mongodb/38024/conf/mongodb.conf
/bin/mongod -f  /mongodb/38025/conf/mongodb.conf
/bin/mongod -f  /mongodb/38026/conf/mongodb.conf
ps -ef |grep mongod
```

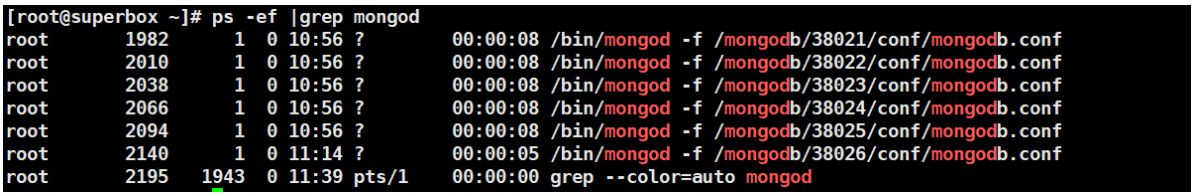


5）如果发现进程：/usr/bin/mongod

```
mongod     1386      1  0 09:10 ?        00:00:21 /usr/bin/mongod -f /etc/mongod.conf
```

可以把这个杀掉，减少内存占用。

```
kill -9 1386
```



6）搭建复制集：

```
mongo --port 38021 admin <<eof
config = {_id: 'sh1', members: [
                           {_id: 0, host:
'127.0.0.1:38021'},
                           {_id: 1, host:
'127.0.0.1:38022'},
                           {_id: 2, host:
'127.0.0.1:38023',"arbiterOnly":true}]
            }
rs.initiate(config)
eof

mongo --port 38024 admin <<eof
```

```
config = {_id: 'sh2', members: [
                                {_id: 0, host:
'127.0.0.1:38024'},
                                {_id: 1, host:
'127.0.0.1:38025'},
                                {_id: 2, host:
'127.0.0.1:38026',"arbiterOnly":true}]
                }
rs.initiate(config)
eof
```

```
[root@superbox ~]# mongo --port 38021 admin <<eof
> config = {_id: 'sh1', members: [
>                         {_id: 0, host: '127.0.0.1:38021'},
>                         {_id: 1, host: '127.0.0.1:38022'},
>                         {_id: 2, host: '127.0.0.1:38023',"arbiterOnly":true}]
>             }
> rs.initiate(config)
> eof
MongoDB shell version v3.6.23
connecting to: mongodb://127.0.0.1:38021/admin?gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("ae7f1a82-adbf-4211-8bcc-e09a50a3366b") }
MongoDB server version: 3.6.23
{
        "_id" : "sh1",
        "members" : [
                {
                        "_id" : 0,
                        "host" : "127.0.0.1:38021"
                },
                {
                        "_id" : 1,
                        "host" : "127.0.0.1:38022"
                },
                {
                        "_id" : 2,
                        "host" : "127.0.0.1:38023",
                        "arbiterOnly" : true
                }
        ]
}
{ "ok" : 1 }
bye
```

```
[root@superbox ~]# mongo --port 38024 admin <<eof
> config = {_id: 'sh2', members: [
>                         {_id: 0, host: '127.0.0.1:38024'},
>                         {_id: 1, host: '127.0.0.1:38025'},
>                         {_id: 2, host: '127.0.0.1:38026',"arbiterOnly":true}]
>             }
> rs.initiate(config)
> eof
MongoDB shell version v3.6.23
connecting to: mongodb://127.0.0.1:38024/admin?gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("a81666dc-b1f9-46d4-a741-ab42b01abd65") }
MongoDB server version: 3.6.23
{
        "_id" : "sh2",
        "members" : [
                {
                        "_id" : 0,
                        "host" : "127.0.0.1:38024"
                },
                {
                        "_id" : 1,
                        "host" : "127.0.0.1:38025"
                },
                {
                        "_id" : 2,
                        "host" : "127.0.0.1:38026",
                        "arbiterOnly" : true
                }
        ]
}
{ "ok" : 1 }
bye
```

3.config节点配置

1）目录创建

```
mkdir -p /mongodb/38018/conf  /mongodb/38018/log /mongodb/38018/data
mkdir -p /mongodb/38019/conf  /mongodb/38019/log /mongodb/38019/data
mkdir -p /mongodb/38020/conf  /mongodb/38020/log /mongodb/38020/data
```

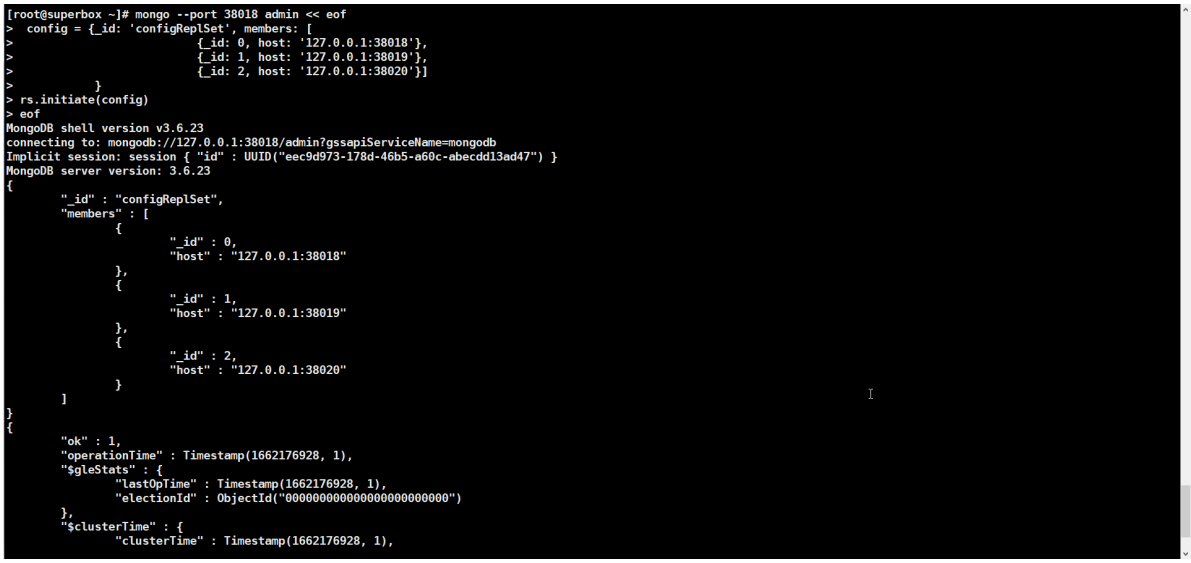
2）修改配置文件：

```
cat > /mongodb/38018/conf/mongodb.conf <<EOF
systemLog:
  destination: file
  path: /mongodb/38018/log/mongodb.conf
  logAppend: true
storage:
  journal:
    enabled: true
  dbPath: /mongodb/38018/data
  directoryPerDB: true
  #engine: wiredTiger
  wiredTiger:
    engineConfig:
      cacheSizeGB: 1
      directoryForIndexes: true
    collectionConfig:
      blockCompressor: zlib
    indexConfig:
      prefixCompression: true
net:
  bindIp: 127.0.0.1
  port: 38018
replication:
  oplogSizeMB: 2048
  replSetName: configReplSet
sharding:
  clusterRole: configsvr
processManagement:
```

```
  fork: true
EOF

\cp /mongodb/38018/conf/mongodb.conf /mongodb/38019/conf/
\cp /mongodb/38018/conf/mongodb.conf /mongodb/38020/conf/

sed 's#38018#38019#g' /mongodb/38019/conf/mongodb.conf -i
sed 's#38018#38020#g' /mongodb/38020/conf/mongodb.conf -i
```

3）启动节点，并配置复制集：

```
/bin/mongod  -f /mongodb/38018/conf/mongodb.conf
/bin/mongod  -f /mongodb/38019/conf/mongodb.conf
/bin/mongod  -f /mongodb/38020/conf/mongodb.conf

# 注：configserver 可以是一个节点，官方建议复制集。configserver不能有arbiter，新版本中，要求必须是复制集。
# 注：mongodb 3.4之后，虽然要求config server为replica set，但是不支持arbiter。

mongo --port 38018 admin << eof
 config = {_id: 'configReplSet', members: [
                           {_id: 0, host: '127.0.0.1:38018'},
                           {_id: 1, host: '127.0.0.1:38019'},
                           {_id: 2, host: '127.0.0.1:38020'}]
            }
rs.initiate(config)
eof
```



5.mongos节点配置:

1）创建目录:

```
mkdir -p /mongodb/38017/conf  /mongodb/38017/log
```

2）配置文件:

```
cat > /mongodb/38017/conf/mongos.conf <<EOF
systemLog:
  destination: file
  path: /mongodb/38017/log/mongos.log
  logAppend: true
net:
  bindIp: 127.0.0.1
  port: 38017
sharding:
  configDB: configReplSet/127.0.0.1:38018,127.0.0.1:38019,127.0.0.1:38020
processManagement:
  fork: true
EOF
```

3）分片集群添加节点

连接到其中一个mongos，做以下配置：

```
# （1）启动mongos
/bin/mongos -f /mongodb/38017/conf/mongos.conf
# （2）连接到mongs的admin数据库
su - mongod
mongo 127.0.0.1:38017/admin
# （3）添加分片
db.runCommand( { addshard : "sh1/127.0.0.1:38021,127.0.0.1:38022,127.0.0.1:38023",name:"shard1"} )
db.runCommand( { addshard : "sh2/127.0.0.1:38024,127.0.0.1:38025,127.0.0.1:38026",name:"shard2"} )
# （4）列出分片
db.runCommand( { listshards : 1 } )
# （5）整体状态查看
sh.status();
```

6.使用分片集群

RANGE分片配置及测试

1.激活数据库分片功能

```
mongo --port 38017 admin
( { enablesharding : "sh1" } )

db.runCommand( { enablesharding : "test" } )
```

2、指定分片键对集合分片

1）创建索引

```
use test
db.vast.ensureIndex( { id: 1 } )
```

2）开启分片

```
use admin
db.runCommand( { shardcollection : "test.vast",key : {id: 1} } )
```

3、集合分片验证

```
admin> use test
for(i=1;i<1000000;i++){ db.vast.insert({"id":i,"name":"shenzheng","age":70,"date":new Date()}); }
db.vast.stats()
```

4、分片结果测试

```
shard1:
mongo --port 38021
db.vast.count();

shard2:
mongo --port 38024
db.vast.count();
```

Hash分片：对weiwei库下的vast大表进行hash

7.创建哈希索引

（1）对于weiwei库开启分片功能

```
mongo --port 38017 admin
use admin
db.runCommand( { enablesharding : "weiwei" } )
```

（2）对于oldliu库下的vast表建立hash索引

```
use weiwei
db.vast.ensureIndex( { id: "hashed" } )
```

(3) 开启分片

```
use admin
sh.shardCollection( "weiwei.vast", { id: "hashed" } )
```

(4) 录入10w行数据测试

```
use weiwei
for(i=1;i<100000;i++){
db.vast.insert({"id":i,"name":"shenzheng","age":70,"date
":new Date()}); }
```

(5) hash分片结果测试

```
mongo --port 38021
use weiwei
db.vast.count();
mongo --port 38024
use weiwei
db.vast.count();
```

8.分片集群的查询及管理

1) 判断是否Shard集群

```
db.runCommand({ isdbgrid : 1})
```

2) 列出所有分片信息

```
db.runCommand({ listshards : 1})
```

3) 列出开启分片的数据库

```
use config
db.databases.find( { "partitioned": true } )
```

或者：

```
db.databases.find() //列出所有数据库分片情况
```

9.查看分片的片键

```
db.collections.find().pretty()
{
    "_id" : "test.vast",
    "lastmodEpoch" :
ObjectId("58a599f19c898bbfb818b63c"),
    "lastmod" : ISODate("1970-02-19T17:02:47.296Z"),
    "dropped" : false,
    "key" : {
        "id" : 1
    },
    "unique" : false
}
```

10.查看分片的详细信息

```
sh.status()
```

11.删除分片节点（谨慎）

（1）确认blance是否在工作

```
sh.getBalancerState()
```

（2）删除shard2节点(谨慎)

```
db.runCommand( { removeShard: "shard2" } )
```

注意：删除操作一定会立即触发blancer。

12.balancer操作

1）介绍

mongos的一个重要功能，自动巡查所有shard节点上的chunk的情况，自动做chunk迁移。

什么时候工作?

> 1、自动运行，会检测系统不繁忙的时候做迁移
> 2、在做节点删除的时候，立即开始迁移工作
> 3、balancer只能在预设定的时间窗口内运行

1）增加新分片：

```
db.runCommand( { addshard : "sh3/127.0.0.1:38021,127.0.0.1:38022,127.0.0.1:38023",name:"shard3"} )
```

2）有需要时可以关闭和开启blancer（备份的时候）

```
sh.stopBalancer()
sh.startBalancer()
```

3）自定义 自动平衡进行的时间段

```
https://docs.mongodb.com/manual/tutorial/manage-sharded-cluster-balancer/#schedule-the-balancing-window
// connect to mongos
use config
sh.setBalancerState( true )
db.settings.update({ _id : "balancer" }, { $set : { activeWindow : { start : "3:00", stop : "5:00" } } }, true )
sh.getBalancerWindow()
sh.status()
```

13.关于集合的balancer（了解下）

1）关闭某个集合的balance

```
sh.disableBalancing("students.grades")
```

2）打开某个集合的balancer

```
sh.enableBalancing("students.grades")
```

3）确定某个集合的balance是开启或者关闭

```
db.getSiblingDB("config").collections.findOne({_id :
"students.grades"}).noBalance;
```