# docker-compose部署mongo分片副本集群（多机）终极版

## 一 相关概念

### 什么是复制集

复制集是一组拥有相同数据的mongodb节点，由两台及以上的mongodb实例组成的集群，包括primary主节点（读写能力）、secondary从节点（读能力）、投票节点3种角色。复制集提供了数据冗余、备份能力，提高了数据的可用性，保证了数据的安全性。

### 启动顺序

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.

### 分片原理

分片集群由3部分组成：

**config server**：配置服务器，存储所有的数据库元数据（分片、路由）信息；可以配置副本；

**shard server：**分片服务器，由一个或多个mongod进程组成，存储数据；可以配置副本；

**router server**：路由服务器，分片集群的入口，所有的请求都由router（mongos）路由到指定的shard服务器上；可以配置副本；

## 二 准备工作

### 1 服务器规划

|  |  |  |  |
| --- | --- | --- | --- |
| **IP** | **192.168.31.51** | **192.168.31.52** | **192.168.31.53** |
| **服务端口** | config-server1:27019 | config-server2:27019 | config-server3:27019 |
| **服务端口** | shard1主节点：17001 | shard1副节点：17002 | shard1仲裁节点：17003 |
| **服务端口** | shard2仲裁节点：27003 | shard2主节点：27001 | shard2副节点：27002 |
| **服务端口** | shard3副节点：37002 | shard3仲裁节点：37003 | shard3主节点：37001 |
| **服务端口** | mongos1:27017 | mongos2:27017 | mongos3:27017 |

说明：shard服务端口释义：第一个数字代表第几分片，最后一个数字代表某个分片的副本角色（1为主节点，2为副节点，3为仲裁节点）。比如17001，表示第一个分片的主节点。

### 2 创建相应目录

目录结构

├── conf

│   ├── config1

│   │   └── config.conf

│   ├── mongos

│   │   └── mongos.conf

│   ├── shard11

│   │   └── shard.conf

│   ├── shard23

│   │   └── shard.conf

│   └── shard32

│   └── shard.conf

├── data

│   ├── config1

│   ├── mongos

│   ├── shard11

│   ├── shard23

│   └── shard32

├── deldir.sh

├── docker-compose.yml

├── key.file

├── log

│   ├── config1

│   ├── mongos

│   ├── shard11

│   ├── shard23

│   └── shard32

└── mkdir.sh

mkdir -pv /data/mongo #3台

进入到/data/mongo 创建相应目录

#!/bin/bash  
mkdir -pv ./{data,log,conf}/{config1,shard11,shard23,shard32,mongos}  
chmod -R a+rwx log

#!/bin/bash  
mkdir -pv ./{data,log,conf}/{config2,shard12,shard21,shard33,mongos}  
chmod -R a+rwx log

#!/bin/bash  
mkdir -pv ./{data,log,conf}/{config3,shard13,shard22,shard31,mongos}  
chmod -R a+rwx log

删除目录

#!/bin/bash  
rm -rf ./log  
rm -rf ./data

### 3 生成keyFile

MongoDB使用keyfile认证，副本集中的每个mongod实例使用keyfile内容作为认证其他成员的共享密码。mongod实例只有拥有正确的keyfile才可以加入副本集。

* keyFile的内容必须是6到1024个字符的长度，且副本集所有成员的keyFile内容必须相同。

* 有一点要注意是的：在UNIX系统中，keyFile必须没有组权限或完全权限（也就是权限要设置成X00的形式）。Windows系统中，keyFile权限没有被检查。

* 可以使用任意方法生成keyFile。例如，如下操作使用openssl生成复杂的随机的1024个字符串。然后使用chmod修改文件权限，只给文件拥有者提供读权限。

openssl rand -base64 756 > key.file  
chmod 400 key.file  
chown 999:999 key.file

999用户是容器中的mongod用户，通过chown修改文件用户权限

### 4 创建docker专用网络

docker network create --subnet=10.20.0.0/24 mongo-network

### 5 docker-compse.yml

#### 5.1 192.168.31.51 docker-compse.yml

version: '3.3'  
services:  
# 配置服务器configsvr  
 mongo-config1:  
 image: mongo:5.0  
 privileged: true  
 container\_name: mongo-config1  
 hostname: mongo-config1  
 restart: always  
 ports:   
 - 27019:27019  
 command: mongod --config /etc/mongod/config.conf --logpath /var/log/mongod/mongodb.log  
 volumes:  
 - /etc/localtime:/etc/localtime:ro  
 #- ${PWD}/data/config1:/data/db  
 - ${PWD}/data/config1:/data  
 - ${PWD}/log/config1:/var/log/mongod  
 - ${PWD}/conf/config1:/etc/mongod  
 - ${PWD}/key.file:/etc/key.file  
 networks:  
 mongo-network:  
 ipv4\_address: 10.20.0.10  
  
# 配置shard1主节点  
 mongo-shard1-1:  
 image: mongo:5.0  
 container\_name: mongo-shard1-1  
 hostname: mongo-shard1-1  
 restart: always  
 ports:  
 - 17001:27018  
 #- 17018:27018  
 command: mongod --config /etc/mongod/shard.conf --logpath /var/log/mongod/shard.log  
 volumes:  
 - /etc/localtime:/etc/localtime:ro  
 #- ${PWD}/data/shard11:/data/db/  
 - ${PWD}/data/shard11:/data  
 - ${PWD}/log/shard11:/var/log/mongod  
 - ${PWD}/conf/shard11:/etc/mongod/  
 - ${PWD}/key.file:/etc/key.file  
 networks:  
 mongo-network:  
 ipv4\_address: 10.20.0.11  
 depends\_on:   
 - mongo-config1  
  
  
# 配置shard2仲裁节点  
 mongo-shard2-3:  
 image: mongo:5.0  
 container\_name: mongo-shard2-3  
 hostname: mongo-shard2-3  
 restart: always  
 ports:  
 - 27003:27018  
 #- 27018:27018  
 command: mongod --config /etc/mongod/shard.conf --logpath /var/log/mongod/shard.log  
 volumes:  
 - /etc/localtime:/etc/localtime:ro  
 #- ${PWD}/data/shard23:/data/db/  
 - ${PWD}/data/shard23:/data  
 - ${PWD}/log/shard23:/var/log/mongod  
 - ${PWD}/conf/shard23:/etc/mongod/  
 - ${PWD}/key.file:/etc/key.file   
 networks:  
 mongo-network:  
 ipv4\_address: 10.20.0.23  
 depends\_on:   
 - mongo-config1  
  
# 配置shard3副节点  
 mongo-shard3-2:  
 image: mongo:5.0  
 container\_name: mongo-shard3-2  
 hostname: mongo-shard3-2  
 restart: always  
 ports:  
 - 37002:27018  
 #- 37018:27018  
 command: mongod --config /etc/mongod/shard.conf --logpath /var/log/mongod/shard.log  
 volumes:  
 - /etc/localtime:/etc/localtime:ro  
 #- ${PWD}/data/shard32:/data/db  
 - ${PWD}/data/shard32:/data  
 - ${PWD}/log/shard32:/var/log/mongod  
 - ${PWD}/conf/shard32:/etc/mongod  
 - ${PWD}/key.file:/etc/key.file  
 networks:  
 mongo-network:  
 ipv4\_address: 10.20.0.32  
 depends\_on:   
 - mongo-config1  
  
#mongos  
 mongos:  
 image: mongo:5.0  
 networks:  
 mongo-network:  
 ipv4\_address: 10.20.0.41  
 container\_name: mongos  
 hostname: mongos  
 command: mongos --config /etc/mongod/mongos.conf --logpath /var/log/mongod/mongos.log  
 #command: mongos --config /etc/mongo/mongos.conf  
 volumes:  
 - /etc/localtime:/etc/localtime:ro  
 - ${PWD}/data/mongos:/data  
 - ${PWD}/log/mongos:/var/log/mongod  
 - ${PWD}/conf/mongos:/etc/mongod  
 - ${PWD}/key.file:/etc/key.file  
 ports:  
 - "27017:27017"  
 depends\_on:  
 - mongo-shard1-1  
 - mongo-shard2-3  
 - mongo-shard3-2  
  
networks:  
 mongo-network:   
 external: true

[📎docker-compose1.yml](https://www.yuque.com/attachments/yuque/0/2022/yml/29027294/1665624876909-7e8d2fde-ba8d-4a68-a4d4-cb66cfff16ff.yml)

#### 5.2 192.168.31.52 docker-compse.yml

version: '3.3'  
networks:   
 mongo-network:  
 external: true  
services:  
# 配置服务器configsvr2  
 mongo-config2:  
 image: mongo:5.0  
 privileged: true  
 container\_name: mongo-config2  
 hostname: mongo-config1  
 restart: always  
 ports:   
 - 27019:27019  
 command: mongod --config /etc/mongod/config.conf --logpath /var/log/mongod/mongodb.log  
 volumes:  
 - /etc/localtime:/etc/localtime:ro  
 #- ${PWD}/data/config2:/data/db  
 #- ${PWD}/data/config2:/data  
 - ${PWD}/log/config2:/var/log/mongod  
 - ${PWD}/conf/config2:/etc/mongod  
 - ${PWD}/key.file:/etc/key.file  
 networks:  
 mongo-network:  
 ipv4\_address: 10.20.0.10  
  
# 配置shard1副节点  
 mongo-shard1-2:  
 image: mongo:5.0  
 container\_name: mongo-shard1-2  
 hostname: mongo-shard1-2  
 restart: always  
 ports:  
 - 17002:27018  
 #- 17018:27018  
 command: mongod --config /etc/mongod/shard.conf --logpath /var/log/mongod/shard.log  
 volumes:  
 - /etc/localtime:/etc/localtime:ro  
 #- ${PWD}/data/shard12:/data/db/  
 - ${PWD}/data/shard12:/data  
 - ${PWD}/log/shard12:/var/log/mongod  
 - ${PWD}/conf/shard12:/etc/mongod/  
 - ${PWD}/key.file:/etc/key.file  
 networks:  
 mongo-network:  
 ipv4\_address: 10.20.0.12  
 depends\_on:   
 - mongo-config2  
  
  
# 配置shard2主节点  
 mongo-shard2-1:  
 image: mongo:5.0  
 container\_name: mongo-shard2-1  
 hostname: mongo-shard2-1  
 restart: always  
 ports:  
 - 27001:27018  
 #- 27018:27018  
 command: mongod --config /etc/mongod/shard.conf --logpath /var/log/mongod/shard.log  
 volumes:  
 - /etc/localtime:/etc/localtime:ro  
 #- ${PWD}/data/shard21:/data/db/  
 - ${PWD}/data/shard21:/data  
 - ${PWD}/log/shard21:/var/log/mongod  
 - ${PWD}/conf/shard21:/etc/mongod/  
 - ${PWD}/key.file:/etc/key.file   
 networks:  
 mongo-network:  
 ipv4\_address: 10.20.0.21  
 depends\_on:   
 - mongo-config2  
  
# 配置shard3仲裁节点  
 mongo-shard3-3:  
 image: mongo:5.0  
 container\_name: mongo-shard3-3  
 hostname: mongo-shard3-3  
 restart: always  
 ports:  
 - 37003:27018  
 #- 37018:27018  
 command: mongod --config /etc/mongod/shard.conf --logpath /var/log/mongod/shard.log  
 volumes:  
 - /etc/localtime:/etc/localtime:ro  
 #- ${PWD}/data/shard33:/data/db  
 - ${PWD}/data/shard33:/data  
 - ${PWD}/log/shard33:/var/log/mongod  
 - ${PWD}/conf/shard33:/etc/mongod  
 - ${PWD}/key.file:/etc/key.file  
 networks:  
 mongo-network:  
 ipv4\_address: 10.20.0.33  
 depends\_on:   
 - mongo-config2  
  
#mongos  
 mongos:  
 image: mongo:5.0  
 networks:  
 mongo-network:  
 ipv4\_address: 10.20.0.42  
 container\_name: mongos  
 hostname: mongos  
 command: mongos --config /etc/mongod/mongos.conf --logpath /var/log/mongod/mongos.log  
 #command: mongos --config /etc/mongo/mongos.conf  
 volumes:  
 - /etc/localtime:/etc/localtime:ro  
 - ${PWD}/data/mongos:/data  
 - ${PWD}/log/mongos:/var/log/mongod  
 - ${PWD}/conf/mongos:/etc/mongod  
 - ${PWD}/key.file:/etc/key.file  
 ports:  
 - "27017:27017"  
 depends\_on:  
 - mongo-shard1-2  
 - mongo-shard2-1  
 - mongo-shard3-3

[📎docker-compose2.yml](https://www.yuque.com/attachments/yuque/0/2022/yml/29027294/1665624884444-c2e3827f-05df-4824-9fcd-1ad4cc5fa13d.yml)

#### 5.3 192.168.31.53 docker-compse.yml

version: '3.3'  
networks:   
 mongo-network:  
 external: true  
services:  
# 配置服务器configsvr3  
 mongo-config3:  
 image: mongo:5.0  
 privileged: true  
 container\_name: mongo-config3  
 hostname: mongo-config3  
 restart: always  
 ports:   
 - 27019:27019  
 command: mongod --config /etc/mongod/config.conf --logpath /var/log/mongod/mongodb.log  
 volumes:  
 - /etc/localtime:/etc/localtime:ro  
 #- ${PWD}/data/config3:/data/db  
 - ${PWD}/data/config3:/data  
 - ${PWD}/log/config3:/var/log/mongod  
 - ${PWD}/conf/config3:/etc/mongod  
 - ${PWD}/key.file:/etc/key.file  
 networks:  
 mongo-network:  
 ipv4\_address: 10.20.0.10  
  
# 配置shard1仲裁节点  
 mongo-shard1-3:  
 image: mongo:5.0  
 container\_name: mongo-shard1-3  
 hostname: mongo-shard1-3  
 restart: always  
 ports:  
 - 17003:27018  
 #- 17018:27018  
 command: mongod --config /etc/mongod/shard.conf --logpath /var/log/mongod/shard.log  
 volumes:  
 - /etc/localtime:/etc/localtime:ro  
 #- ${PWD}/data/shard13:/data/db/  
 - ${PWD}/data/shard13:/data  
 - ${PWD}/log/shard13:/var/log/mongod  
 - ${PWD}/conf/shard13:/etc/mongod/  
 - ${PWD}/key.file:/etc/key.file  
 networks:  
 mongo-network:  
 ipv4\_address: 10.20.0.11  
 depends\_on:   
 - mongo-config3  
  
  
# 配置shard2副节点  
 mongo-shard2-2:  
 image: mongo:5.0  
 container\_name: mongo-shard2-2  
 hostname: mongo-shard2-2  
 restart: always  
 ports:  
 - 27002:27018  
 #- 27018:27018  
 command: mongod --config /etc/mongod/shard.conf --logpath /var/log/mongod/shard.log  
 volumes:  
 - /etc/localtime:/etc/localtime:ro  
 #- ${PWD}/data/shard22:/data/db/  
 - ${PWD}/data/shard22:/data  
 - ${PWD}/log/shard22:/var/log/mongod  
 - ${PWD}/conf/shard22:/etc/mongod/  
 - ${PWD}/key.file:/etc/key.file   
 networks:  
 mongo-network:  
 ipv4\_address: 10.20.0.22  
 depends\_on:   
 - mongo-config3  
  
# 配置shard3主节点  
 mongo-shard3-1:  
 image: mongo:5.0  
 container\_name: mongo-shard3-1  
 hostname: mongo-shard3-1  
 restart: always  
 ports:  
 - 37001:27018  
 #- 37018:27018  
 command: mongod --config /etc/mongod/shard.conf --logpath /var/log/mongod/shard.log  
 volumes:  
 - /etc/localtime:/etc/localtime:ro  
 #- ${PWD}/data/shard31:/data/db  
 - ${PWD}/data/shard31:/data  
 - ${PWD}/log/shard31:/var/log/mongod  
 - ${PWD}/conf/shard31:/etc/mongod  
 - ${PWD}/key.file:/etc/key.file  
 networks:  
 mongo-network:  
 ipv4\_address: 10.20.0.31  
 depends\_on:   
 - mongo-config3  
  
#mongos  
 mongos:  
 image: mongo:5.0  
 networks:  
 mongo-network:  
 ipv4\_address: 10.20.0.43  
 container\_name: mongos  
 hostname: mongos  
 command: mongos --config /etc/mongod/mongos.conf --logpath /var/log/mongod/mongos.log  
 #command: mongos --config /etc/mongo/mongos.conf  
 volumes:  
 - /etc/localtime:/etc/localtime:ro  
 - ${PWD}/data/mongos:/data  
 - ${PWD}/log/mongos/:/var/log/mongod  
 - ${PWD}/conf/mongos/:/etc/mongod  
 - ${PWD}/key.file:/etc/key.file  
 ports:  
 - "27017:27017"  
 depends\_on:  
 - mongo-shard1-3  
 - mongo-shard2-2  
 - mongo-shard3-1

[📎docker-compose3.yml](https://www.yuque.com/attachments/yuque/0/2022/yml/29027294/1665624891948-a21c2019-4410-46bf-82f2-daa6168bc696.yml)

## 三 部署配置服务器

### 1 配置文件config.conf(3台都要配置，内容一样)

sharding:  
 clusterRole: configsvr # 定义为mongo配置服务器  
  
replication:  
 replSetName: cfgsvr # 副本集名称，相同副本须使用同一个副本集名称  
  
storage:  
 dbPath: "/data/db"  
 wiredTiger:  
 engineConfig:  
 journalCompressor: "zstd"  
 directoryForIndexes: true  
 collectionConfig:  
 blockCompressor: "zstd"  
  
security:  
 authorization: enabled  
 clusterAuthMode: "keyFile"  
 keyFile: "/etc/key.file"  
  
systemLog:  
 verbosity: 0  
 quiet: false  
 traceAllExceptions: false  
 destination: "file"  
 logAppend: true  
 logRotate: reopen  
  
net:  
 bindIpAll: true  
 port: 27019

# 关于存储引擎wiredTiger的参数设置

# 参考：<https://docs.mongodb.com/manual/reference/configuration-options/#storage-options>

<https://zhuanlan.zhihu.com/p/78333051?utm_source=qq>

zstd

MongoDB 对数据的压缩支持 snappy、zstd 和 zlib 算法,新版本的MONGODB 提供了关于journal压缩的 zstd 模式，这个压缩的方式比 zlib 压缩的方式要更节省CPU资源，这点做得要比其他数据库在压缩方面具有优势。

directoryForIndexes: true

//默认false索引集合storage.dbPath存储在数据单独子目录，这里必须配置为true，将数据和索引分开存储。否则所有库的数据文件都会存放在一个目录中.

### 2 启动服务config\_server

docker-compose up -d mongo-config1  
docker-compose up -d mongo-config2  
docker-compose up -d mongo-config3

### 3 初始化配置服务复制集

# 登录mongo，创建的三个配置服务中随便一个

docker exec -it mongo-config1 mongo 127.0.0.1:27019  
------------------------  
use admin  
rs.initiate({  
 \_id: "cfgsvr",  
 configsvr: true,  
 members: [  
 { \_id : 0, host : "192.168.31.51:27019" },  
 { \_id : 1, host : "192.168.31.52:27019" },  
 { \_id : 2, host : "192.168.31.53:27019" }  
 ]  
 }  
)

## 

### 4 查看状态

rs\_configsvr:PRIMARY> rs.status()

{

"set" : "cfgsvr",

"date" : ISODate("2022-10-12T12:10:25.206Z"),

"myState" : 1,

"term" : NumberLong(1),

"syncSourceHost" : "",

"syncSourceId" : -1,

"configsvr" : true,

"heartbeatIntervalMillis" : NumberLong(2000),

"majorityVoteCount" : 2,

"writeMajorityCount" : 2,

"votingMembersCount" : 3,

"writableVotingMembersCount" : 3,

"optimes" : {

"lastCommittedOpTime" : {

"ts" : Timestamp(1665576624, 1),

"t" : NumberLong(1)

},

"lastCommittedWallTime" : ISODate("2022-10-12T12:10:24.515Z"),

"readConcernMajorityOpTime" : {

"ts" : Timestamp(1665576624, 1),

"t" : NumberLong(1)

},

"appliedOpTime" : {

"ts" : Timestamp(1665576624, 1),

"t" : NumberLong(1)

},

"durableOpTime" : {

"ts" : Timestamp(1665576624, 1),

"t" : NumberLong(1)

},

"lastAppliedWallTime" : ISODate("2022-10-12T12:10:24.515Z"),

"lastDurableWallTime" : ISODate("2022-10-12T12:10:24.515Z")

},

"lastStableRecoveryTimestamp" : Timestamp(1665576621, 1),

"electionCandidateMetrics" : {

"lastElectionReason" : "electionTimeout",

"lastElectionDate" : ISODate("2022-10-12T12:09:33.222Z"),

"electionTerm" : NumberLong(1),

"lastCommittedOpTimeAtElection" : {

"ts" : Timestamp(1665576562, 1),

"t" : NumberLong(-1)

},

"lastSeenOpTimeAtElection" : {

"ts" : Timestamp(1665576562, 1),

"t" : NumberLong(-1)

},

"numVotesNeeded" : 2,

"priorityAtElection" : 1,

"electionTimeoutMillis" : NumberLong(10000),

"numCatchUpOps" : NumberLong(0),

"newTermStartDate" : ISODate("2022-10-12T12:09:33.271Z"),

"wMajorityWriteAvailabilityDate" : ISODate("2022-10-12T12:09:34.032Z")

},

"members" : [

{

"\_id" : 0,

"name" : "192.168.31.51:27019",

"health" : 1,

"state" : 1,

"stateStr" : "PRIMARY",

"uptime" : 638,

"optime" : {

"ts" : Timestamp(1665576624, 1),

"t" : NumberLong(1)

},

"optimeDate" : ISODate("2022-10-12T12:10:24Z"),

"lastAppliedWallTime" : ISODate("2022-10-12T12:10:24.515Z"),

"lastDurableWallTime" : ISODate("2022-10-12T12:10:24.515Z"),

"syncSourceHost" : "",

"syncSourceId" : -1,

"infoMessage" : "",

"electionTime" : Timestamp(1665576573, 1),

"electionDate" : ISODate("2022-10-12T12:09:33Z"),

"configVersion" : 1,

"configTerm" : 1,

"self" : true,

"lastHeartbeatMessage" : ""

},

{

"\_id" : 1,

"name" : "192.168.31.52:27019",

"health" : 1,

"state" : 2,

"stateStr" : "SECONDARY",

"uptime" : 62,

"optime" : {

"ts" : Timestamp(1665576622, 1),

"t" : NumberLong(1)

},

"optimeDurable" : {

"ts" : Timestamp(1665576622, 1),

"t" : NumberLong(1)

},

"optimeDate" : ISODate("2022-10-12T12:10:22Z"),

"optimeDurableDate" : ISODate("2022-10-12T12:10:22Z"),

"lastAppliedWallTime" : ISODate("2022-10-12T12:10:24.515Z"),

"lastDurableWallTime" : ISODate("2022-10-12T12:10:24.515Z"),

"lastHeartbeat" : ISODate("2022-10-12T12:10:23.285Z"),

"lastHeartbeatRecv" : ISODate("2022-10-12T12:10:24.606Z"),

"pingMs" : NumberLong(0),

"lastHeartbeatMessage" : "",

"syncSourceHost" : "192.168.31.51:27019",

"syncSourceId" : 0,

"infoMessage" : "",

"configVersion" : 1,

"configTerm" : 1

},

{

"\_id" : 2,

"name" : "192.168.31.53:27019",

"health" : 1,

"state" : 2,

"stateStr" : "SECONDARY",

"uptime" : 62,

"optime" : {

"ts" : Timestamp(1665576622, 1),

"t" : NumberLong(1)

},

"optimeDurable" : {

"ts" : Timestamp(1665576622, 1),

"t" : NumberLong(1)

},

"optimeDate" : ISODate("2022-10-12T12:10:22Z"),

"optimeDurableDate" : ISODate("2022-10-12T12:10:22Z"),

"lastAppliedWallTime" : ISODate("2022-10-12T12:10:24.515Z"),

"lastDurableWallTime" : ISODate("2022-10-12T12:10:24.515Z"),

"lastHeartbeat" : ISODate("2022-10-12T12:10:23.285Z"),

"lastHeartbeatRecv" : ISODate("2022-10-12T12:10:23.941Z"),

"pingMs" : NumberLong(0),

"lastHeartbeatMessage" : "",

"syncSourceHost" : "192.168.31.51:27019",

"syncSourceId" : 0,

"infoMessage" : "",

"configVersion" : 1,

"configTerm" : 1

}

],

"ok" : 1,

"$gleStats" : {

"lastOpTime" : Timestamp(1665576562, 1),

"electionId" : ObjectId("7fffffff0000000000000001")

},

"lastCommittedOpTime" : Timestamp(1665576624, 1),

"$clusterTime" : {

"clusterTime" : Timestamp(1665576624, 1),

"signature" : {

"hash" : BinData(0,"5/Fxwbj/mULzy5viIEQzzVLgQBM="),

"keyId" : NumberLong("7153596910018756631")

}

},

"operationTime" : Timestamp(1665576624, 1)

}

## 四 创建分片副本集

### shard1分片副本集

#### 1、配置文件shard1.conf(3台配置内容一样)

**一个副本集里的三个节点的三个配置文件里都必须一样**

sharding:  
 clusterRole: shardsvr # 配置为分片服务  
replication:  
 replSetName: shardsvr1 # 分片服务名称,3个分片名称不一致  
  
storage:  
 dbPath: "/data/db" #分片数据库路径  
 wiredTiger:  
 engineConfig:  
 journalCompressor: "zstd"  
 directoryForIndexes: true  
 collectionConfig:  
 blockCompressor: "zstd"  
  
security:  
 authorization: enabled  
 clusterAuthMode: "keyFile"  
 keyFile: "/etc/key.file"  
  
systemLog:  
 verbosity: 0  
 quiet: false  
 traceAllExceptions: false  
 destination: "file"  
 logAppend: true  
 logRotate: reopen  
  
net:  
 bindIpAll: true  
 port: 27018

#### 2、shard1分片部署

docker-compose up -d mongo-shard1-1  
docker-compose up -d mongo-shard1-2  
docker-compose up -d mongo-shard1-3

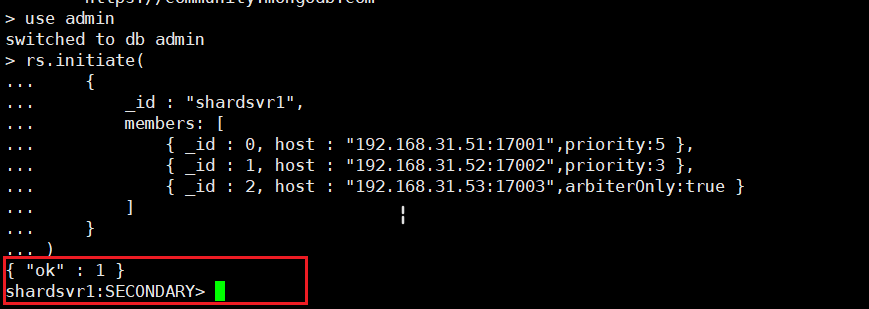
#### 3、初始化分片服务器

登录进容器中，任意一台

docker exec -it mongo-shard1-1 mongo 127.0.0.1:27018  
-----------------  
use admin  
rs.initiate(  
 {  
 \_id : "shardsvr1",  
 members: [  
 { \_id : 0, host : "192.168.31.51:17001",priority:5 },  
 { \_id : 1, host : "192.168.31.52:17002",priority:3 },  
 { \_id : 2, host : "192.168.31.53:17003",arbiterOnly:true }  
 ]  
 }  
)

设置裁决节点只需要在任意一个节点后增加arbiterOnly:true即可

指定设置主副节点，使用priority，设置优先级，值大的优先设置主节点



#### 4、查看状态

rs.status()

shardsvr1:SECONDARY> rs.status()

{

"set" : "shardsvr1",

"date" : ISODate("2022-10-12T12:20:03.686Z"),

"myState" : 1,

"term" : NumberLong(1),

"syncSourceHost" : "",

"syncSourceId" : -1,

"heartbeatIntervalMillis" : NumberLong(2000),

"majorityVoteCount" : 2,

"writeMajorityCount" : 2,

"votingMembersCount" : 3,

"writableVotingMembersCount" : 2,

.........略............................

"members" : [

{

"\_id" : 0,

"name" : "192.168.31.51:17001",

"health" : 1,

"state" : 1,

"stateStr" : "PRIMARY",

"uptime" : 178,

"optime" : {

"ts" : Timestamp(1665577200, 1),

"t" : NumberLong(1)

},

.........略............................

"self" : true,

"lastHeartbeatMessage" : ""

},

{

"\_id" : 1,

"name" : "192.168.31.52:17002",

"health" : 1,

"state" : 2,

"stateStr" : "SECONDARY",

"uptime" : 33,

"optime" : {

"ts" : Timestamp(1665577200, 1),

"t" : NumberLong(1)

},

"optimeDurable" : {

"ts" : Timestamp(1665577200, 1),

"t" : NumberLong(1)

},

"optimeDate" : ISODate("2022-10-12T12:20:00Z"),

"optimeDurableDate" : ISODate("2022-10-12T12:20:00Z"),

"lastAppliedWallTime" : ISODate("2022-10-12T12:20:00.511Z"),

"lastDurableWallTime" : ISODate("2022-10-12T12:20:00.511Z"),

"lastHeartbeat" : ISODate("2022-10-12T12:20:02.450Z"),

"lastHeartbeatRecv" : ISODate("2022-10-12T12:20:03.456Z"),

"pingMs" : NumberLong(3),

"lastHeartbeatMessage" : "",

"syncSourceHost" : "192.168.31.51:17001",

"syncSourceId" : 0,

"infoMessage" : "",

"configVersion" : 1,

"configTerm" : 1

},

{

"\_id" : 2,

"name" : "192.168.31.53:17003",

"health" : 1,

"state" : 7,

"stateStr" : "ARBITER",

"uptime" : 33,

"lastHeartbeat" : ISODate("2022-10-12T12:20:02.444Z"),

"lastHeartbeatRecv" : ISODate("2022-10-12T12:20:02.506Z"),

"pingMs" : NumberLong(0),

"lastHeartbeatMessage" : "",

"syncSourceHost" : "",

"syncSourceId" : -1,

"infoMessage" : "",

"configVersion" : 1,

"configTerm" : 1

}

],

"ok" : 1

}

### shard2分片副本集

#### 1 配置文件shard2.conf

**一个副本集里的三个节点的三个配置文件里都必须一样**

sharding:  
 clusterRole: shardsvr # 配置为分片服务  
replication:  
 replSetName: shardsvr2 # 分片服务名称,3个分片名称不一致  
  
storage:  
 dbPath: "/data/db" #分片数据库路径  
 wiredTiger:  
 engineConfig:  
 journalCompressor: "zstd"  
 directoryForIndexes: true  
 collectionConfig:  
 blockCompressor: "zstd"  
  
security:  
 authorization: enabled  
 clusterAuthMode: "keyFile"  
 keyFile: "/etc/key.file"  
  
systemLog:  
 verbosity: 0  
 quiet: false  
 traceAllExceptions: false  
 destination: "file"  
 logAppend: true  
 logRotate: reopen  
  
net:  
 bindIpAll: true  
 port: 27018

#### 2、shard2分片部署

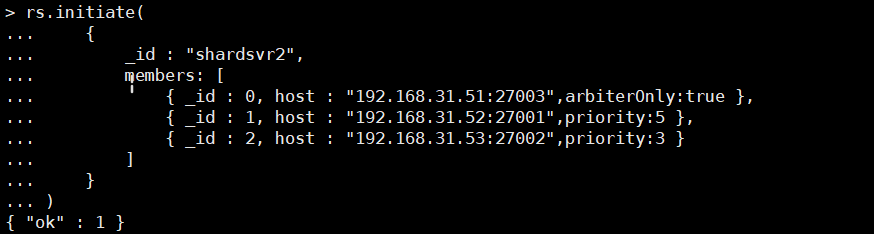
docker-compose up -d mongo-shard2-1  
docker-compose up -d mongo-shard2-2  
docker-compose up -d mongo-shard2-3

#### 3、初始化分片服务器

登录进容器中，任意一台

docker exec -it mongo-shard2-1 mongo 127.0.0.1:27018

use admin  
rs.initiate(  
 {  
 \_id : "shardsvr2",  
 members: [  
 { \_id : 0, host : "192.168.31.51:27003",arbiterOnly:true },  
 { \_id : 1, host : "192.168.31.52:27001",priority:5 },  
 { \_id : 2, host : "192.168.31.53:27002",priority:3 }  
 ]  
 }  
)



### shard3分片副本集

#### 1 配置文件shard3.conf

**一个副本集里的三个节点的三个配置文件里都必须一样**

sharding:  
 clusterRole: shardsvr # 配置为分片服务  
replication:  
 replSetName: shardsvr3 # 分片服务名称,3个分片名称不一致  
  
storage:  
 dbPath: "/data/db" #分片数据库路径  
 wiredTiger:  
 engineConfig:  
 journalCompressor: "zstd"  
 directoryForIndexes: true  
 collectionConfig:  
 blockCompressor: "zstd"  
  
security:  
 authorization: enabled  
 clusterAuthMode: "keyFile"  
 keyFile: "/etc/key.file"  
  
systemLog:  
 verbosity: 0  
 quiet: false  
 traceAllExceptions: false  
 destination: "file"  
 logAppend: true  
 logRotate: reopen  
  
net:  
 bindIpAll: true  
 port: 27018

#### 2 shard3分片部署

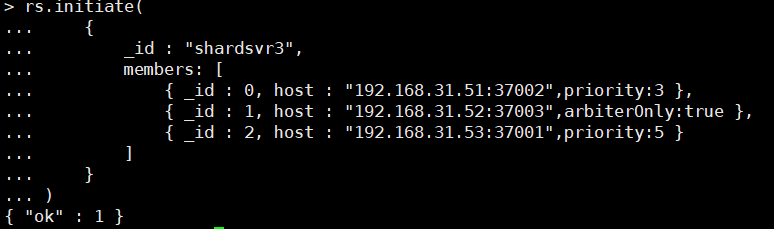
docker-compose up -d mongo-shard3-1  
docker-compose up -d mongo-shard3-2  
docker-compose up -d mongo-shard3-3

#### 3、初始化分片服务器

登录进容器中，任意一台

docker exec -it mongo-shard3-1 mongo 127.0.0.1:27018

use admin  
rs.initiate(  
 {  
 \_id : "shardsvr3",  
 members: [  
 { \_id : 0, host : "192.168.31.51:37002",priority:3 },  
 { \_id : 1, host : "192.168.31.52:37003",arbiterOnly:true },  
 { \_id : 2, host : "192.168.31.53:37001",priority:5 }  
 ]  
 }  
)



## 五、 配置mongos路由服务器

**先启动配置服务器和分片服务器,后启动路由实例**

### 1 配置文件mongos.conf

systemLog:  
 verbosity: 0  
 quiet: false  
 traceAllExceptions: true  
 destination: "file"  
 logAppend: true  
 logRotate: reopen  
  
security:  
 clusterAuthMode: "keyFile"  
 keyFile: "/etc/key.file"  
 # authorization: disabled #该配置项不支持mongos，仅支持mongod  
  
net:  
 bindIpAll: true  
 port: 27017  
  
sharding:  
 #定义为mongos配置服务器 #监听的配置服务器,只能有1个或者3个 rs\_configsvr为配置服务器的副本集名字  
 configDB: cfgsvr/192.168.31.51:27019,192.168.31.52:27019,192.168.31.52:27019

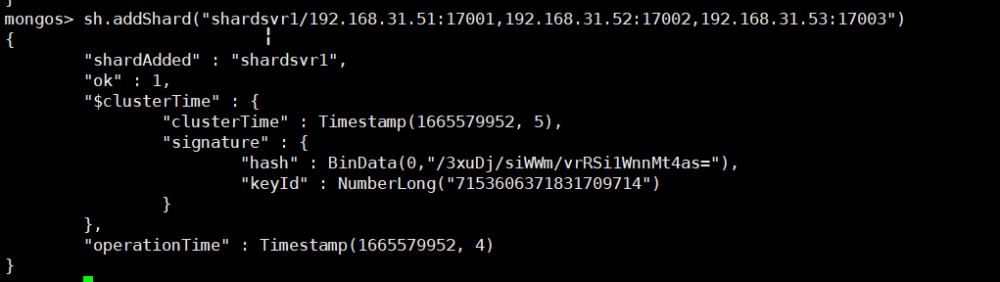
### 1 启动服务

docker-compose up -d mongos

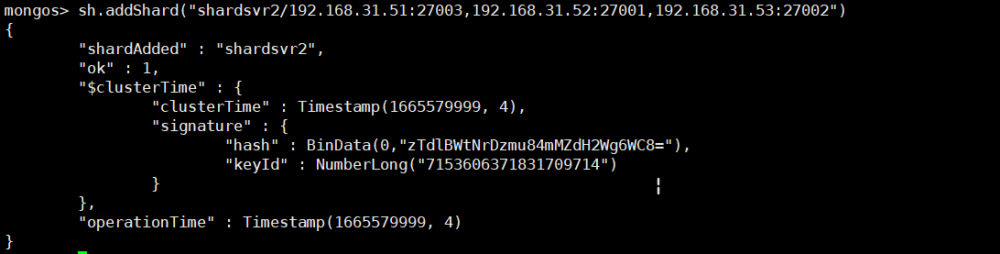
### 2 初始化mongos

docker exec -it mongos mongo 127.0.0.1:27017  
mongos> use admin

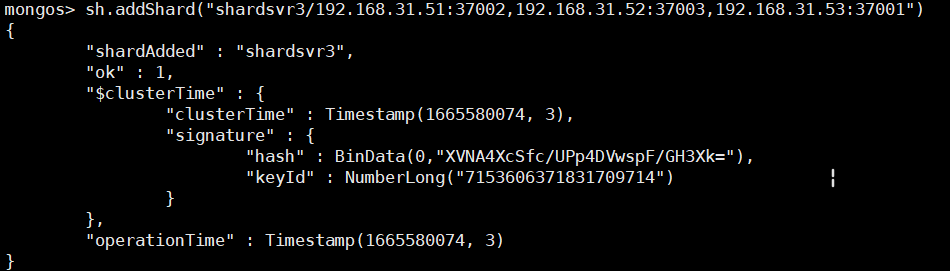
mongos> sh.addShard("shardsvr1/192.168.31.51:17001,192.168.31.52:17002,192.168.31.53:17003")



mongos> sh.addShard("shardsvr2/192.168.31.51:27003,192.168.31.52:27001,192.168.31.53:27002")



mongos> sh.addShard("shardsvr3/192.168.31.51:37002,192.168.31.52:37003,192.168.31.53:37001")



## 六 创建管理员账号

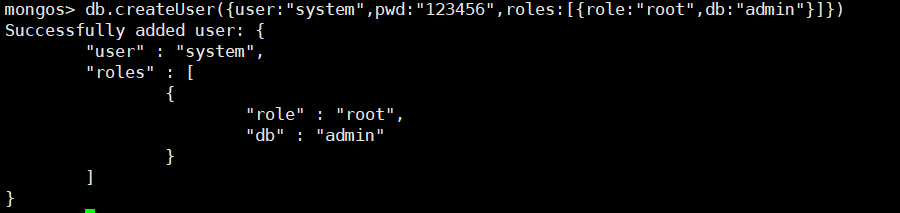
**mongos shard1副本集 shard2副本集 shard3副本集 四个角色都要各自创建账号**

添加两个管理员账号,一个系统管理员:system 一个数据库管理员:administrator

docker exec -it mongos mongo 127.0.0.1:27017

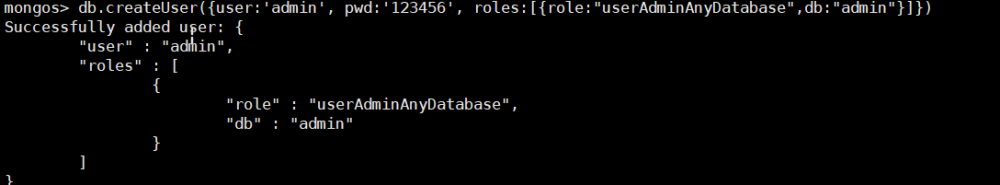
＃先添加系统管理员账号,用来管理用户

mongos> db.createUser({user:"system",pwd:"123456",roles:[{role:"root",db:"admin"}]})  
mongos> sh.status()



# 添加数据库管理员,用来管理所有数据库

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



# 添加管理员用户认证,认证之后才能管理所有数据库

db.auth('system','123456')  
db.auth('admin','123456')

# 退出，用刚才创建的账号进行登录(两个都测试一下)

docker exec -it mongos mongo 127.0.0.1:27017 -u system -p123456 --authenticationDatabase admin  
docker exec -it mongos mongo 127.0.0.1:27017 -u admin -p123456 --authenticationDatabase admin

**shard1**

docker exec -it mongo-shard1-1 mongo 127.0.0.1:27018  
shardsvr1:PRIMARY> use admin  
shardsvr1:PRIMARY> db.createUser({user:"system",pwd:"123456",roles:[{role:"root",db:"admin"}]})  
shardsvr1:PRIMARY> db.auth('system','123456')

docker exec -it mongo-shard1-1 mongo 127.0.0.1:27018 -u system -p123456 --authenticationDatabase admin

**shard2**

docker exec -it mongo-shard2-1 mongo 127.0.0.1:27018  
shardsvr2:PRIMARY> use admin  
shardsvr2:PRIMARY> db.createUser({user:"system",pwd:"123456",roles:[{role:"root",db:"admin"}]})  
shardsvr2:PRIMARY> db.auth('system','123456')

docker exec -it mongo-shard2-1 mongo 127.0.0.1:27018 -u system -p123456 --authenticationDatabase admin

**shard3**

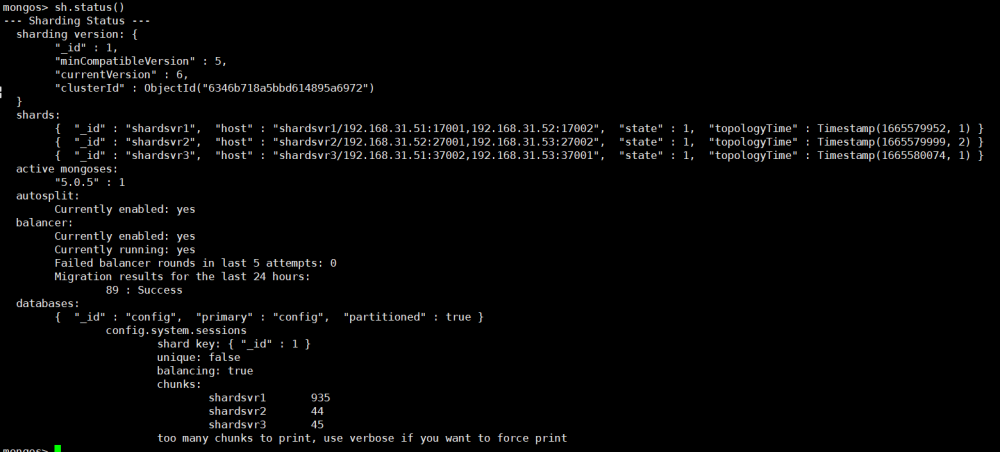
docker exec -it mongo-shard3-1 mongo 127.0.0.1:27018  
shardsvr3:PRIMARY> use admin  
shardsvr3:PRIMARY> db.createUser({user:"system",pwd:"123456",roles:[{role:"root",db:"admin"}]})  
shardsvr3:PRIMARY> db.auth('system','123456')

docker exec -it mongo-shard3-1 mongo 127.0.0.1:27018 -u system -p123456 --authenticationDatabase admin

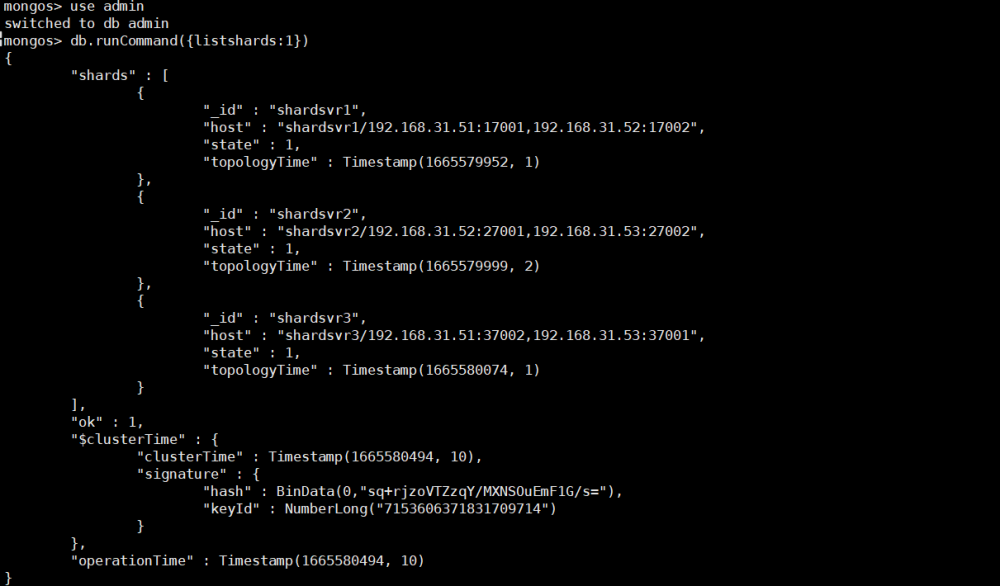
## 七 集群查询

docker exec -it mongos mongo 127.0.0.1:27017 -u system -p123456 --authenticationDatabase admin

### 1 查看集群状态sh.status()或db.runCommand({listshards:1})



db.runCommand({listshards:1})需要先切换到admin库

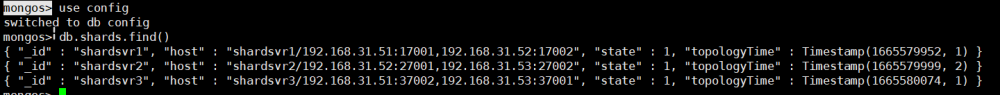


### 2 查看集群间的连接情况

mongos> db.adminCommand("connPoolStats");

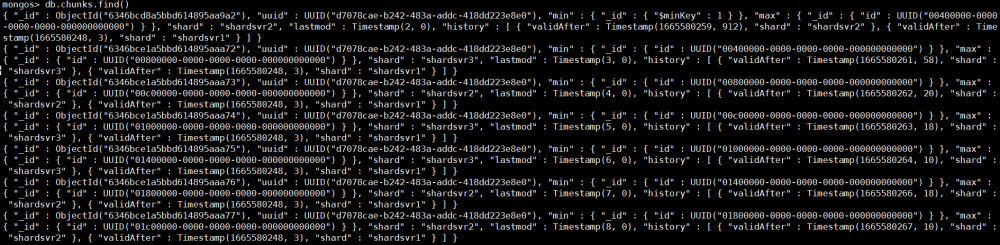
### 3 通过命令查看mongodb路由服务器上的shards集合会有数据展示

mongos> use config  
mongos> db.shards.find()



### 4 通过命令查看mongodb路由服务器上的chunks集合会有数据展示

mongos> use config  
mongos> db.chunks.find()



### 5 查看副本集配置信息：

rs.conf()

docker exec -it mongo-shard1-1 mongo 127.0.0.1:27018  
shardsvr1:PRIMARY> rs.conf()

shardsvr1:PRIMARY> rs.conf()

{

"\_id" : "shardsvr1",

"version" : 2,

"term" : 1,

"members" : [

{

"\_id" : 0,

"host" : "192.168.31.51:17001",

"arbiterOnly" : false,

"buildIndexes" : true,

"hidden" : false,

"priority" : 5,

"tags" : {

},

"secondaryDelaySecs" : NumberLong(0),

"votes" : 1

},

{

"\_id" : 1,

"host" : "192.168.31.52:17002",

"arbiterOnly" : false,

"buildIndexes" : true,

"hidden" : false,

"priority" : 3,

"tags" : {

},

"secondaryDelaySecs" : NumberLong(0),

"votes" : 1

},

{

"\_id" : 2,

"host" : "192.168.31.53:17003",

"arbiterOnly" : true,

"buildIndexes" : true,

"hidden" : false,

"priority" : 0,

"tags" : {

},

"secondaryDelaySecs" : NumberLong(0),

"votes" : 1

}

],

"protocolVersion" : NumberLong(1),

"writeConcernMajorityJournalDefault" : true,

"settings" : {

"chainingAllowed" : true,

"heartbeatIntervalMillis" : 2000,

"heartbeatTimeoutSecs" : 10,

"electionTimeoutMillis" : 10000,

"catchUpTimeoutMillis" : -1,

"catchUpTakeoverDelayMillis" : 30000,

"getLastErrorModes" : {

},

"getLastErrorDefaults" : {

"w" : 1,

"wtimeout" : 0

},

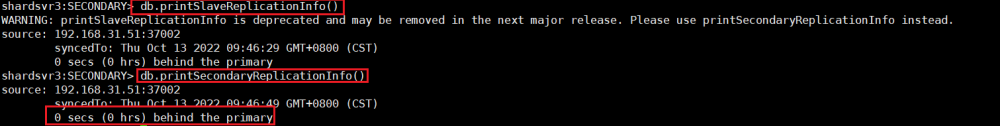
"replicaSetId" : ObjectId("6346b7399a62d0aecec05427")

}

}

### 6 查看同步情况

db.printSlaveReplicationInfo() //查看同步情况  
db.printSecondaryReplicationInfo() //新版采用



## 八 数据验证

### 测试一(数据不会分散到分片上有待验证)

#### 1 连接mongos

docker exec -it mongos mongo 127.0.0.1:27017 -u system -p 123456 --authenticationDatabase admin

#### 3 启动分片

虽然数据库采用分片集群的方式部署，但如果db和collection不启用分片的话（默认是不启用的），数据不会分片存储，此时如果向集群中导入一个db，会将整个db随机存储到任意一个分片中，而不是拆分存储到多个分片。

# enableSharding只能针对admin数据库运行，适宜选用adminuse admin

#指定要分片的数据库  
sh.enableSharding("springboot")  
#指定集合的分片规则  
#这里表示指定springboot库下的user集合的\_id字段（也就是主键，每个集合都有这个字段）按hash散列进行分片，{ id : 1 }表示按字段id进度范围分片，这里id必须是整型  
#要分片存储的集合都需要指定分片规则，分片规则一经创建不可修改，只能删除集合再重新设置  
sh.shardCollection("springboot.user", { \_id : "hashed" } )  
#查询user的集合状态  
db.user.stats()  
# 5、测试分片效果。  
use springboot  
# 尝试写入数据，并观察数据分块  
# 插入5百万个简单的文档，耐心等待插入结束,可以少一点。1000改为10  
for(var i=1;i<=1000;i++){  
 db.user.insert({  
 name:i,  
 age:Math.round(Math.random() \* 100),  
 score1:Math.round(Math.random() \* 100),  
 score2:Math.round(Math.random() \* 100),  
 score3:Math.round(Math.random() \* 100),  
 score4:Math.round(Math.random() \* 100),  
 score5:Math.round(Math.random() \* 100)  
 });  
}  
#查询user的集合状态  
db.user.stats()

### 测试二

#### 1 插入数据

docker exec -it mongos mongo 127.0.0.1:27017 -u system -p 123456 --authenticationDatabase admin

db.runCommand({"enablesharding":"testdb2"})  
db.runCommand({"shardcollection":"testdb2.person","key":{\_id:'hashed'}})  
use testdb2  
for(var i=0;i<30;i++){db.person.insert({name:"testdata"+i});}

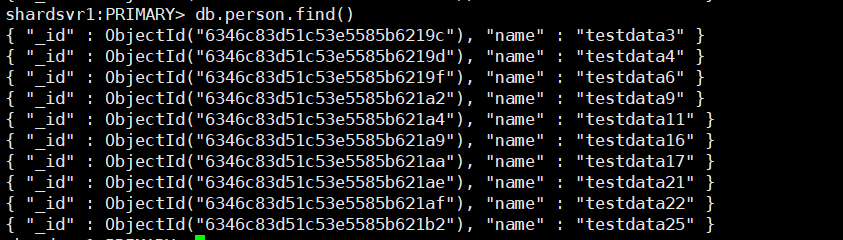
#### 2 查看数据是否已经分片

##### 1 shard1主节点查看

docker exec -it mongo-shard1-1 mongo 127.0.0.1:27018 -u system -p123456 --authenticationDatabase admin

**接下来在查看分片情况**

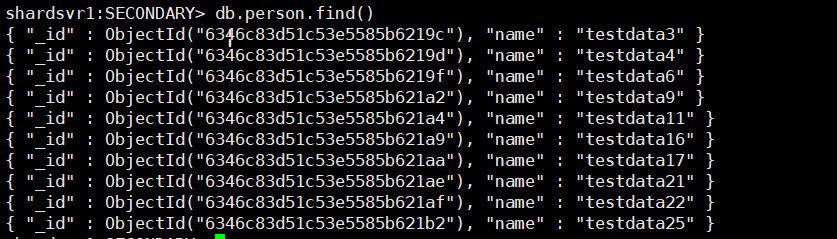
shardsvr1:PRIMARY> db.auth("system","123456")  
shardsvr1:PRIMARY> use testdb2  
shardsvr1:PRIMARY> db.person.find()



##### 2 shard1副节点查看

docker exec -it mongo-shard1-2 mongo 127.0.0.1:27018 -u system -p 123456 --authenticationDatabase admin

shardsvr1:SECONDARY> db.getMongo().setSecondaryOk();db.getMongo().setSecondaryOk()  
shardsvr1:SECONDARY> use testdb2  
shardsvr1:SECONDARY> db.person.find()

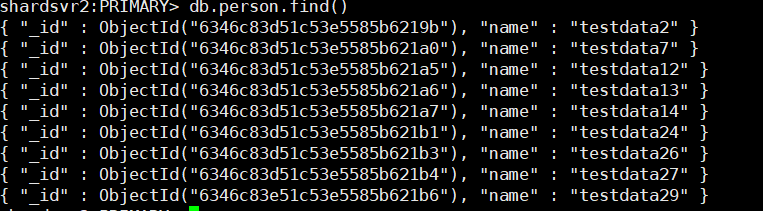


##### 3 shard2主节点查看

docker exec -it mongo-shard2-1 mongo 127.0.0.1:27018 -u system -p123456 --authenticationDatabase admin

**接下来在查看分片情况**

shardsvr2:PRIMARY> use testdb2  
shardsvr2:PRIMARY> db.person.find()

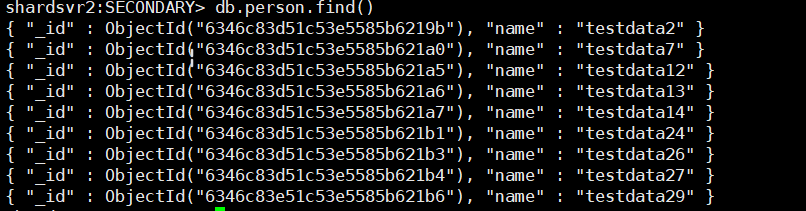


##### 4 shard2副节点查看

docker exec -it mongo-shard2-2 mongo 127.0.0.1:27018 -u system -p123456 --authenticationDatabase admin

**接下来在查看分片情况**

shardsvr2:SECONDARY> db.getMongo().setSecondaryOk();db.getMongo().setSecondaryOk()  
shardsvr2:SECONDARY> use testdb2  
shardsvr2:SECONDARY> db.person.find()

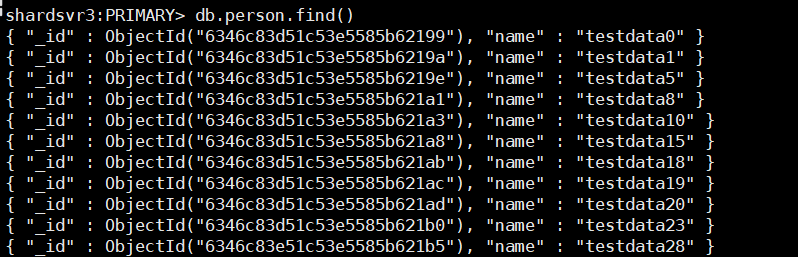


##### 5 shard3主节点查看

docker exec -it mongo-shard3-1 mongo 127.0.0.1:27018 -u system -p123456 --authenticationDatabase admin

**接下来在查看分片情况**

shardsvr3:PRIMARY> db.auth("system","123456")  
shardsvr3:PRIMARY> use testdb2  
shardsvr3:PRIMARY> db.person.find()

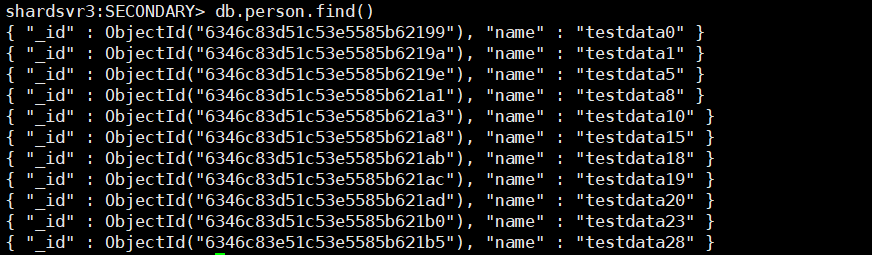


##### 6 shard3副节点查看

docker exec -it mongo-shard3-2 mongo 127.0.0.1:27018 -u system -p123456 --authenticationDatabase admin

**接下来在查看分片情况**

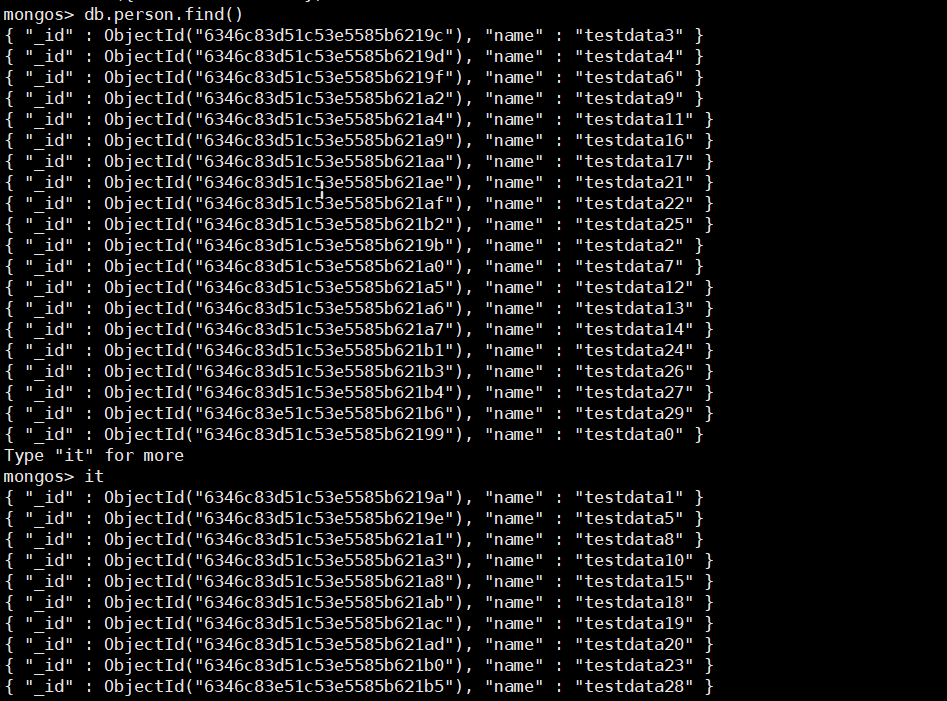
shardsvr3:SECONDARY> db.getMongo().setSecondaryOk();db.getMongo().setSecondaryOk()  
shardsvr3:SECONDARY> use testdb2  
shardsvr3:SECONDARY> db.person.find()



##### 4 mongos查看

可以查看到所有数据

docker exec -it mongos mongo 127.0.0.1:27017 -u system -p 123456 --authenticationDatabase admin



## 九 模拟故障

### 1 模拟shard1 主节点故障

#### 1.1 停掉shard1 主节点mongo-shard1-1

docker-compose stop mongo-shard1-1

此时连接shard1副节点

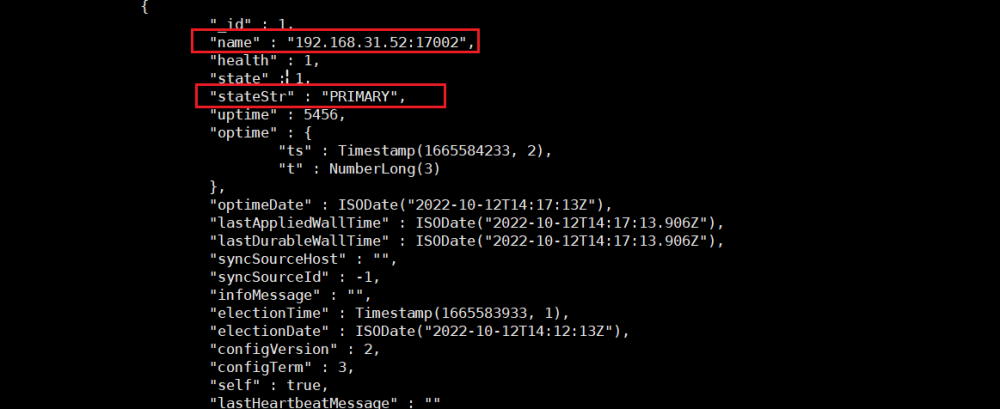
docker exec -it mongo-shard1-2 mongo 127.0.0.1:27018 -u system -p 123456 --authenticationDatabase admin

发现由原来的SECONDARY变成了PRIMARY

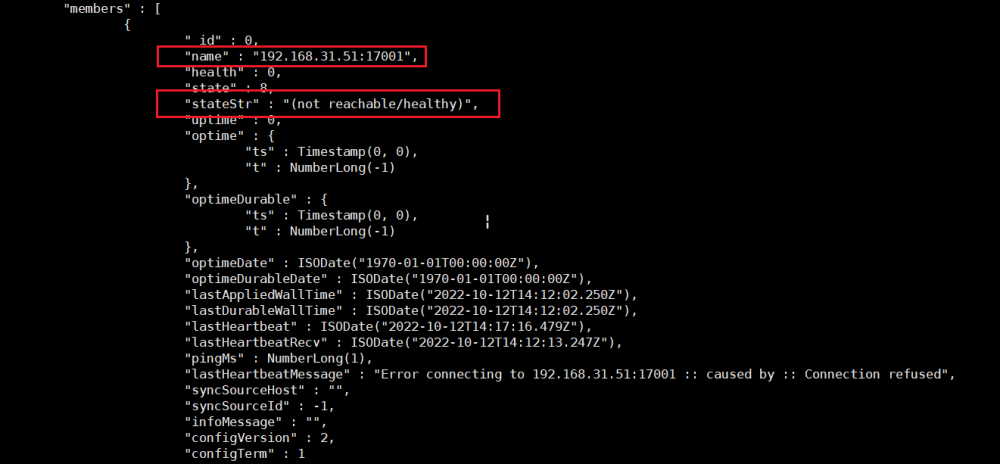


或者通过查看副本集状态也可得知副节点变成了主节点

rs.status()



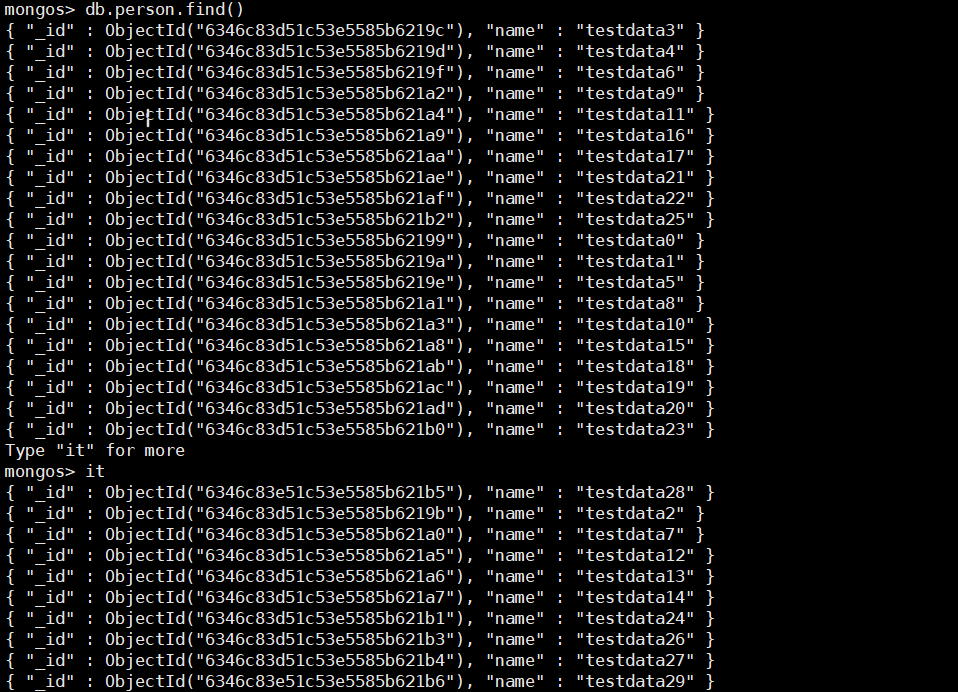
而主节点为not reachable/healthy 状态



mongos查询数据也是完整的

mongos>use testdb2

mongos> db.person.find()

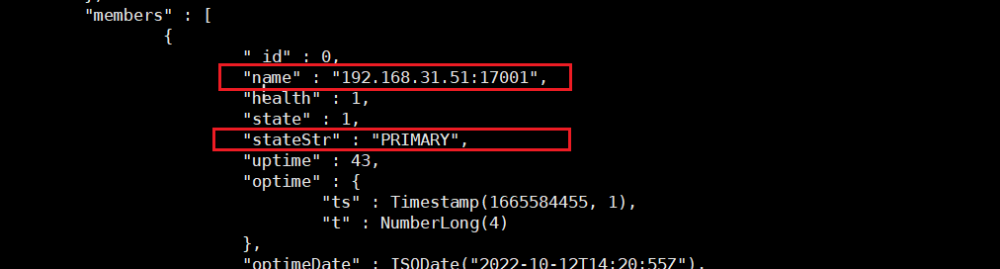


#### 1.2 把shard1主节点恢复起来

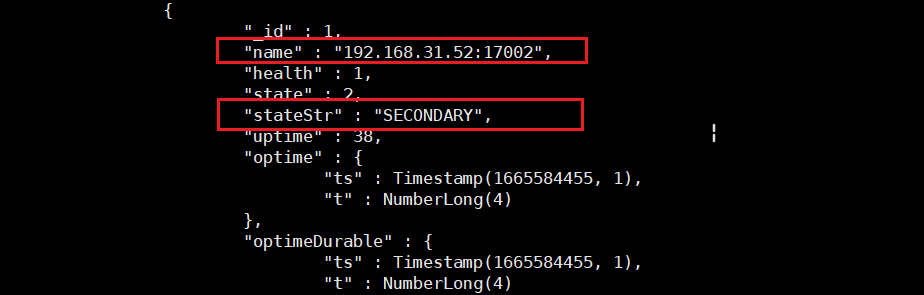
docker-compose start mongo-shard1-1

docker exec -it mongo-shard1-1 mongo 127.0.0.1:27018 -u system -p 123456 --authenticationDatabase admin

主节点还是成为主节点



副节点变回原来的角色副节点

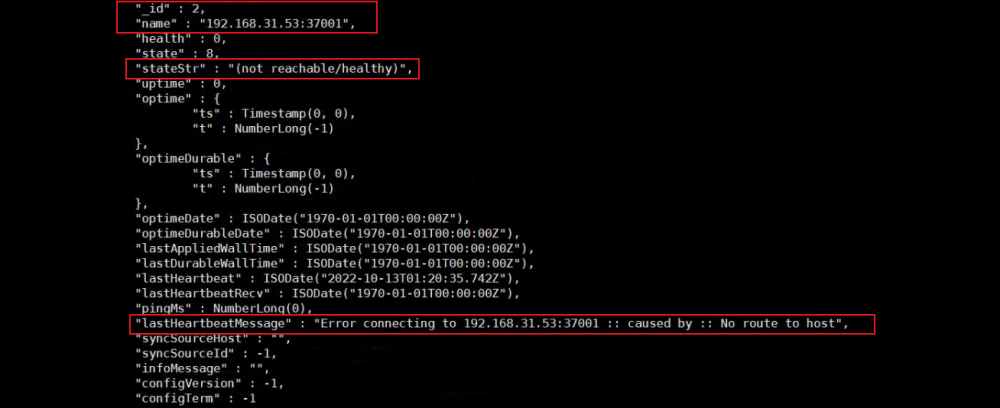


### 2 关机第3台服务器192.168.31.53

此时第三台服务器无法正常运作，mongos查询数据正常。

docker exec -it mongo-shard3-2 mongo 127.0.0.1:27018 -u system -p123456 --authenticationDatabase admin

rs.status()



## 十 其他

#### 增长mongo0的权重：

cfg = rs.conf()

# 修改权重

cfg.members[0].priority=5

# 从新配置

rs.reconfig(cfg)

仲裁节点的权重默认为 0，其它节点默认为 1。调高mongo0节点的权重后，若是mongo0宕机，mongo1会成为新主节点，当mongo0恢复后会重新成为主节点。

#### 验证副本集

切换节点查看同步状态：

rs.printReplicationInfo()

仅当建立了集合后副节点才会进行同步。

#### 备份恢复

##### 备份

#备份所有库

docker exec -it mongos mongodump -h 127.0.0.1:27017 -u system -p123456 --authenticationDatabase admin -o /data/backup/mongoDB

##### 恢复

--nsInclude: 参数恢复整个数据库或某个数据集(新版本的指令可以使用)

--noIndexRestore参数：恢复数据时不创建索引

#恢复testdb2库

docker exec -it mongos mongorestore -h 127.0.0.1:27017 -u system -p123456 --authenticationDatabase admin --noIndexRestore --nsInclude="testdb2.\*" /data/backup/mongoDB/

# 恢复 testdb2.person数据集

docker exec -it mongos mongorestore -h 127.0.0.1:27017 -u system -p123456 --authenticationDatabase admin --noIndexRestore --nsInclude="testdb2.person" /data/backup/mongoDB/

## 参考文档

<https://blog.csdn.net/qq_39637094/article/details/123448604> docker-compose创建mongodb分片集群

<https://zhuanlan.zhihu.com/p/135351892>

<https://www.136.la/mysql/show-71320.html> docker-compose搭建mongoDB副本集（1主+1副+1仲裁）

<https://www.jb51.cc/docker/1043755.html> docker-compose搭建mongoDB副本集（1主+1副+1仲裁）

<https://www.icode9.com/content-4-1343633.html> docker-compose 部署mongo副本集集群，主从仲裁

<https://blog.csdn.net/qq_32096997/article/details/115012436> 没有副本

<https://www.minws.com/archives/626/> 强制配置

<https://blog.csdn.net/qq_42554735/article/details/108792359> 非docker