# docker-compose部署mongo分片副本集群(单机)

## 一 准备工作

### 1 创建相应目录

目录结构

conf/

├── config

│   └── config.conf

├── mongos

│   └── mongos.conf

├── shard1

│   └── shard.conf

└── shard2

└── shard.conf

├── data

│   ├── config1

│   ├── config2

│   ├── config3

│   ├── mongos

│   ├── shard11

│   ├── shard12

│   ├── shard13

│   ├── shard21

│   ├── shard22

│   └── shard23

└── log

├── config1

├── config2

├── config3

├── mongos

├── shard11

├── shard12

├── shard13

├── shard21

├── shard22

└── shard23

mkdir -pv /data/mongo  
mkdir -pv ./{data,log}/{config1,config2,config3,shard11,shard12,shard13,shard21,shard22,shard23,mongos}  
mkdir -pv ./conf/{config,shard1,shard2,mongos}  
chmod -R a+rwx log

### 2 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

### 3 创建docker专用网络

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

### 4 角色端口规划

|  |  |  |  |
| --- | --- | --- | --- |
| 服务端口 | mongos:27017 |  |  |
| 服务端口 | configserver:27019 | configserver:27019 | configserver:27019 |
| 服务端口 | shard1 主节点:17001  shard2 主节点:27003 | shard1 副节点:17002  shard1 副节点:27002 | shard1 仲裁节点:17003  shard1 仲裁节点:27001 |

说明：端口五位数，第二位表示分片号。第3位表示角色编号。

### 5 docker-compose.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}/log/config1:/var/log/mongod  
 - ${PWD}/conf/config:/etc/mongod  
 - ${PWD}/key.file:/etc/key.file  
 networks:  
 mongo-network:  
 ipv4\_address: 10.20.0.11  
  
 mongo-config2:  
 image: mongo:5.0  
 privileged: true  
 container\_name: mongo-config2  
 hostname: mongo-config2  
 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}/log/config2:/var/log/mongod  
 - ${PWD}/conf/config:/etc/mongod  
 - ${PWD}/key.file:/etc/key.file  
 networks:  
 mongo-network:  
 ipv4\_address: 10.20.0.12  
  
 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}/log/config3:/var/log/mongod  
 - ${PWD}/conf/config:/etc/mongod  
 - ${PWD}/key.file:/etc/key.file  
 networks:  
 mongo-network:  
 ipv4\_address: 10.20.0.13  
  
  
# 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}/log/shard11:/var/log/mongod  
 - ${PWD}/conf/shard1:/etc/mongod/  
 - ${PWD}/key.file:/etc/key.file  
 networks:  
 mongo-network:  
 ipv4\_address: 10.20.0.21  
 depends\_on:   
 - mongo-config1  
 - mongo-config2  
 - mongo-config3  
  
# 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}/log/shard12:/var/log/mongod  
 - ${PWD}/conf/shard1:/etc/mongod/  
 - ${PWD}/key.file:/etc/key.file  
 networks:  
 mongo-network:  
 ipv4\_address: 10.20.0.22  
 depends\_on:  
 - mongo-config1  
 - mongo-config2  
 - mongo-config3  
  
# 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}/log/shard13:/var/log/mongod  
 - ${PWD}/conf/shard1:/etc/mongod/  
 - ${PWD}/key.file:/etc/key.file  
 networks:  
 mongo-network:  
 ipv4\_address: 10.20.0.23  
 depends\_on:  
 - mongo-config1  
 - mongo-config2  
 - mongo-config3  
  
  
# 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}/log/shard21:/var/log/mongod  
 - ${PWD}/conf/shard2:/etc/mongod/  
 - ${PWD}/key.file:/etc/key.file   
 networks:  
 mongo-network:  
 ipv4\_address: 10.20.0.31  
 depends\_on:   
 - mongo-config1  
 - mongo-config2  
 - 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}/log/shard22:/var/log/mongod  
 - ${PWD}/conf/shard2:/etc/mongod/  
 - ${PWD}/key.file:/etc/key.file  
 networks:  
 mongo-network:  
 ipv4\_address: 10.20.0.32  
 depends\_on:  
 - mongo-config1  
 - mongo-config2  
 - mongo-config3  
  
# 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}/log/shard23:/var/log/mongod  
 - ${PWD}/conf/shard2:/etc/mongod/  
 - ${PWD}/key.file:/etc/key.file  
 networks:  
 mongo-network:  
 ipv4\_address: 10.20.0.33  
 depends\_on:  
 - mongo-config1  
 - mongo-config2  
 - mongo-config3  
  
#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}/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-shard1-2  
 - mongo-shard1-3  
 - mongo-shard2-1  
 - mongo-shard2-2  
 - mongo-shard2-3  
  
networks:  
 mongo-network:   
 external: true

## 二 部署配置服务器

### 1 配置文件config.conf

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

### 2 启动服务config

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

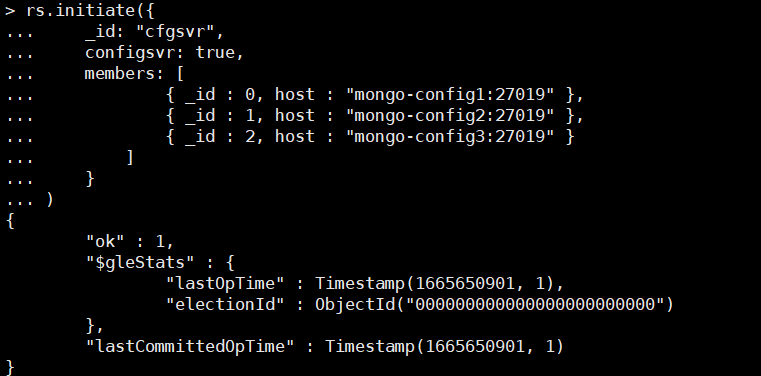
### 3 初始化配置服务复制集

# 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

# 初始化

use admin  
rs.initiate({  
 \_id: "cfgsvr",  
 configsvr: true,  
 members: [  
 { \_id : 0, host : "mongo-config1:27019" },  
 { \_id : 1, host : "mongo-config2:27019" },  
 { \_id : 2, host : "mongo-config3:27019" }  
 ]  
 }  
)



查看状态

rs\_configsvr:PRIMARY> rs.status()

cfgsvr:PRIMARY> rs.status()

{

"set" : "cfgsvr",

"date" : ISODate("2022-10-13T11:41:46.072Z"),

"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(1665661305, 2),

"t" : NumberLong(1)

},

"lastCommittedWallTime" : ISODate("2022-10-13T11:41:45.805Z"),

"readConcernMajorityOpTime" : {

"ts" : Timestamp(1665661305, 2),

"t" : NumberLong(1)

},

"appliedOpTime" : {

"ts" : Timestamp(1665661305, 2),

"t" : NumberLong(1)

},

"durableOpTime" : {

"ts" : Timestamp(1665661305, 2),

"t" : NumberLong(1)

},

"lastAppliedWallTime" : ISODate("2022-10-13T11:41:45.805Z"),

"lastDurableWallTime" : ISODate("2022-10-13T11:41:45.805Z")

},

"lastStableRecoveryTimestamp" : Timestamp(1665661293, 1),

"electionCandidateMetrics" : {

"lastElectionReason" : "electionTimeout",

"lastElectionDate" : ISODate("2022-10-13T08:48:31.963Z"),

"electionTerm" : NumberLong(1),

"lastCommittedOpTimeAtElection" : {

"ts" : Timestamp(1665650901, 1),

"t" : NumberLong(-1)

},

"lastSeenOpTimeAtElection" : {

"ts" : Timestamp(1665650901, 1),

"t" : NumberLong(-1)

},

"numVotesNeeded" : 2,

"priorityAtElection" : 1,

"electionTimeoutMillis" : NumberLong(10000),

"numCatchUpOps" : NumberLong(0),

"newTermStartDate" : ISODate("2022-10-13T08:48:32.044Z"),

"wMajorityWriteAvailabilityDate" : ISODate("2022-10-13T08:48:32.337Z")

},

"members" : [

{

"\_id" : 0,

"name" : "mongo-config1:27019",

"health" : 1,

"state" : 1,

"stateStr" : "PRIMARY",

"uptime" : 13031,

"optime" : {

"ts" : Timestamp(1665661305, 2),

"t" : NumberLong(1)

},

"optimeDate" : ISODate("2022-10-13T11:41:45Z"),

"lastAppliedWallTime" : ISODate("2022-10-13T11:41:45.805Z"),

"lastDurableWallTime" : ISODate("2022-10-13T11:41:45.805Z"),

"syncSourceHost" : "",

"syncSourceId" : -1,

"infoMessage" : "",

"electionTime" : Timestamp(1665650911, 1),

"electionDate" : ISODate("2022-10-13T08:48:31Z"),

"configVersion" : 1,

"configTerm" : 1,

"self" : true,

"lastHeartbeatMessage" : ""

},

{

"\_id" : 1,

"name" : "mongo-config2:27019",

"health" : 1,

"state" : 2,

"stateStr" : "SECONDARY",

"uptime" : 10404,

"optime" : {

"ts" : Timestamp(1665661304, 1),

"t" : NumberLong(1)

},

"optimeDurable" : {

"ts" : Timestamp(1665661304, 1),

"t" : NumberLong(1)

},

"optimeDate" : ISODate("2022-10-13T11:41:44Z"),

"optimeDurableDate" : ISODate("2022-10-13T11:41:44Z"),

"lastAppliedWallTime" : ISODate("2022-10-13T11:41:45.805Z"),

"lastDurableWallTime" : ISODate("2022-10-13T11:41:45.805Z"),

"lastHeartbeat" : ISODate("2022-10-13T11:41:44.877Z"),

"lastHeartbeatRecv" : ISODate("2022-10-13T11:41:44.771Z"),

"pingMs" : NumberLong(0),

"lastHeartbeatMessage" : "",

"syncSourceHost" : "mongo-config1:27019",

"syncSourceId" : 0,

"infoMessage" : "",

"configVersion" : 1,

"configTerm" : 1

},

{

"\_id" : 2,

"name" : "mongo-config3:27019",

"health" : 1,

"state" : 2,

"stateStr" : "SECONDARY",

"uptime" : 10404,

"optime" : {

"ts" : Timestamp(1665661304, 1),

"t" : NumberLong(1)

},

"optimeDurable" : {

"ts" : Timestamp(1665661304, 1),

"t" : NumberLong(1)

},

"optimeDate" : ISODate("2022-10-13T11:41:44Z"),

"optimeDurableDate" : ISODate("2022-10-13T11:41:44Z"),

"lastAppliedWallTime" : ISODate("2022-10-13T11:41:45.805Z"),

"lastDurableWallTime" : ISODate("2022-10-13T11:41:45.805Z"),

"lastHeartbeat" : ISODate("2022-10-13T11:41:44.891Z"),

"lastHeartbeatRecv" : ISODate("2022-10-13T11:41:44.603Z"),

"pingMs" : NumberLong(0),

"lastHeartbeatMessage" : "",

"syncSourceHost" : "mongo-config1:27019",

"syncSourceId" : 0,

"infoMessage" : "",

"configVersion" : 1,

"configTerm" : 1

}

],

"ok" : 1,

"$gleStats" : {

"lastOpTime" : Timestamp(0, 0),

"electionId" : ObjectId("7fffffff0000000000000001")

},

"lastCommittedOpTime" : Timestamp(1665661305, 2),

"$clusterTime" : {

"clusterTime" : Timestamp(1665661305, 2),

"signature" : {

"hash" : BinData(0,"7JIHsYrt6tGs1q6Chgmpfua91II="),

"keyId" : NumberLong("7153916193592573972")

}

},

"operationTime" : Timestamp(1665661305, 2)

}

## 三 创建分片副本集

### shard1分片副本集

#### 1、配置文件shard1.conf

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 mongo-shard1-2 mongo-shard1-3

#### 3、初始化分片服务器

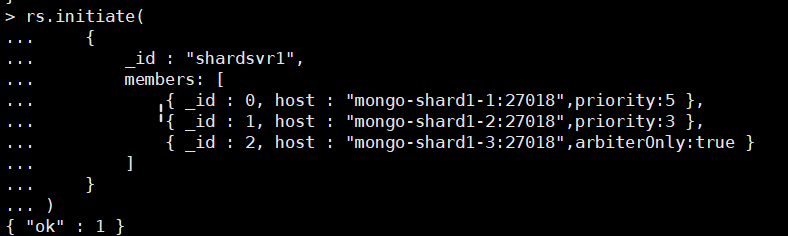
登录进容器中，任意一台

docker exec -it mongo-shard1-1 mongo 127.0.0.1:27018

use admin  
rs.initiate(  
 {  
 \_id : "shardsvr1",  
 members: [  
 { \_id : 0, host : "mongo-shard1-1:27018",priority:5 },  
 { \_id : 1, host : "mongo-shard1-2:27018",priority:3 },  
 { \_id : 2, host : "mongo-shard1-3:27018",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:PRIMARY> rs.status()

{

"set" : "shardsvr1",

"date" : ISODate("2022-10-13T11:19:52.655Z"),

"myState" : 1,

"term" : NumberLong(1),

"syncSourceHost" : "",

"syncSourceId" : -1,

"heartbeatIntervalMillis" : NumberLong(2000),

"majorityVoteCount" : 2,

"writeMajorityCount" : 2,

"votingMembersCount" : 3,

"writableVotingMembersCount" : 2,

"optimes" : {

"lastCommittedOpTime" : {

"ts" : Timestamp(1665659991, 1),

"t" : NumberLong(1)

},

"lastCommittedWallTime" : ISODate("2022-10-13T11:19:51.057Z"),

"readConcernMajorityOpTime" : {

"ts" : Timestamp(1665659991, 1),

"t" : NumberLong(1)

},

"appliedOpTime" : {

"ts" : Timestamp(1665659991, 1),

"t" : NumberLong(1)

},

"durableOpTime" : {

"ts" : Timestamp(1665659991, 1),

"t" : NumberLong(1)

},

"lastAppliedWallTime" : ISODate("2022-10-13T11:19:51.057Z"),

"lastDurableWallTime" : ISODate("2022-10-13T11:19:51.057Z")

},

"lastStableRecoveryTimestamp" : Timestamp(1665659961, 1),

"electionCandidateMetrics" : {

"lastElectionReason" : "electionTimeout",

"lastElectionDate" : ISODate("2022-10-13T08:58:29.959Z"),

"electionTerm" : NumberLong(1),

"lastCommittedOpTimeAtElection" : {

"ts" : Timestamp(1665651499, 1),

"t" : NumberLong(-1)

},

"lastSeenOpTimeAtElection" : {

"ts" : Timestamp(1665651499, 1),

"t" : NumberLong(-1)

},

"numVotesNeeded" : 2,

"priorityAtElection" : 5,

"electionTimeoutMillis" : NumberLong(10000),

"numCatchUpOps" : NumberLong(0),

"newTermStartDate" : ISODate("2022-10-13T08:58:30.133Z"),

"wMajorityWriteAvailabilityDate" : ISODate("2022-10-13T08:58:31.029Z")

},

"members" : [

{

"\_id" : 0,

"name" : "mongo-shard1-1:27018",

"health" : 1,

"state" : 1,

"stateStr" : "PRIMARY",

"uptime" : 8649,

"optime" : {

"ts" : Timestamp(1665659991, 1),

"t" : NumberLong(1)

},

"optimeDate" : ISODate("2022-10-13T11:19:51Z"),

"lastAppliedWallTime" : ISODate("2022-10-13T11:19:51.057Z"),

"lastDurableWallTime" : ISODate("2022-10-13T11:19:51.057Z"),

"syncSourceHost" : "",

"syncSourceId" : -1,

"infoMessage" : "",

"electionTime" : Timestamp(1665651509, 1),

"electionDate" : ISODate("2022-10-13T08:58:29Z"),

"configVersion" : 2,

"configTerm" : 1,

"self" : true,

"lastHeartbeatMessage" : ""

},

{

"\_id" : 1,

"name" : "mongo-shard1-2:27018",

"health" : 1,

"state" : 2,

"stateStr" : "SECONDARY",

"uptime" : 8493,

"optime" : {

"ts" : Timestamp(1665659991, 1),

"t" : NumberLong(1)

},

"optimeDurable" : {

"ts" : Timestamp(1665659991, 1),

"t" : NumberLong(1)

},

"optimeDate" : ISODate("2022-10-13T11:19:51Z"),

"optimeDurableDate" : ISODate("2022-10-13T11:19:51Z"),

"lastAppliedWallTime" : ISODate("2022-10-13T11:19:51.057Z"),

"lastDurableWallTime" : ISODate("2022-10-13T11:19:51.057Z"),

"lastHeartbeat" : ISODate("2022-10-13T11:19:51.251Z"),

"lastHeartbeatRecv" : ISODate("2022-10-13T11:19:51.766Z"),

"pingMs" : NumberLong(0),

"lastHeartbeatMessage" : "",

"syncSourceHost" : "mongo-shard1-1:27018",

"syncSourceId" : 0,

"infoMessage" : "",

"configVersion" : 2,

"configTerm" : 1

},

{

"\_id" : 2,

"name" : "mongo-shard1-3:27018",

"health" : 1,

"state" : 7,

"stateStr" : "ARBITER",

"uptime" : 8493,

"lastHeartbeat" : ISODate("2022-10-13T11:19:51.589Z"),

"lastHeartbeatRecv" : ISODate("2022-10-13T11:19:51.584Z"),

"pingMs" : NumberLong(0),

"lastHeartbeatMessage" : "",

"syncSourceHost" : "",

"syncSourceId" : -1,

"infoMessage" : "",

"configVersion" : 2,

"configTerm" : 1

}

],

"ok" : 1,

"$gleStats" : {

"lastOpTime" : Timestamp(0, 0),

"electionId" : ObjectId("7fffffff0000000000000001")

},

"lastCommittedOpTime" : Timestamp(1665659991, 1),

"$configServerState" : {

"opTime" : {

"ts" : Timestamp(1665659986, 1),

"t" : NumberLong(-1)

}

},

"$clusterTime" : {

"clusterTime" : Timestamp(1665659991, 1),

"signature" : {

"hash" : BinData(0,"VWU4zVp3RJDbpyYDql9acPJsZUg="),

"keyId" : NumberLong("7153916193592573972")

}

},

"operationTime" : Timestamp(1665659991, 1)

}

### shard2分片副本集

#### 1 配置文件shard2.conf

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

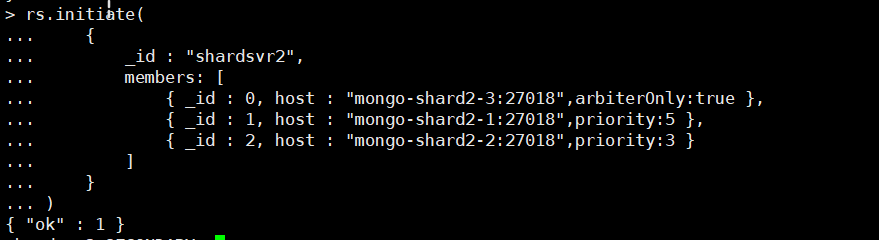
docker-compose up -d mongo-shard2-1 mongo-shard2-2 mongo-shard2-3

#### 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

rs.initiate(  
 {  
 \_id : "shardsvr2",  
 members: [  
 { \_id : 0, host : "mongo-shard2-3:27018",arbiterOnly:true },  
 { \_id : 1, host : "mongo-shard2-1:27018",priority:5 },  
 { \_id : 2, host : "mongo-shard2-2:27018",priority:3 }  
 ]  
 }  
)



## 四、 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/mongo-config1:27019,mongo-config2:27019,mongo-config3:27019

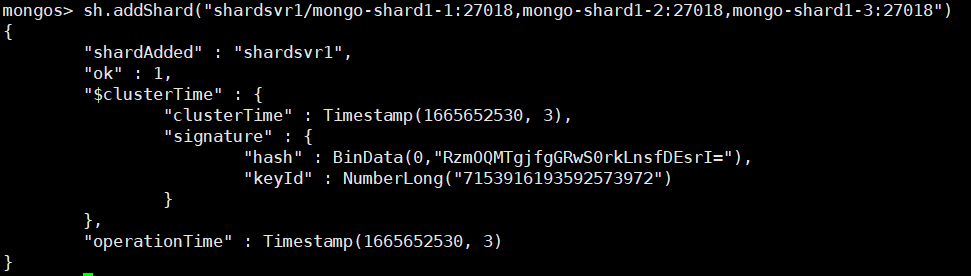
### 1 启动服务

docker-compose up -d mongos

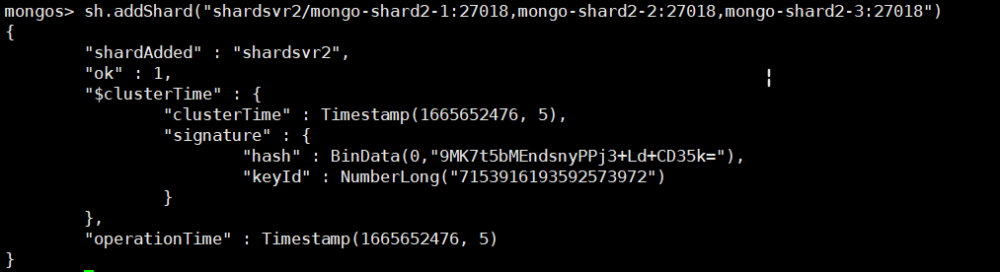
### 2 初始化mongos,将分片加入到集群

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

sh.addShard("shardsvr1/mongo-shard1-1:27018,mongo-shard1-2:27018,mongo-shard1-3:27018")



sh.addShard("shardsvr2/mongo-shard2-1:27018,mongo-shard2-2:27018,mongo-shard2-3:27018")



## 六 创建管理员账号

<https://www.yuque.com/caikefusiji-ydknd/exgu5x/afzwtq>

## 七 集群查询

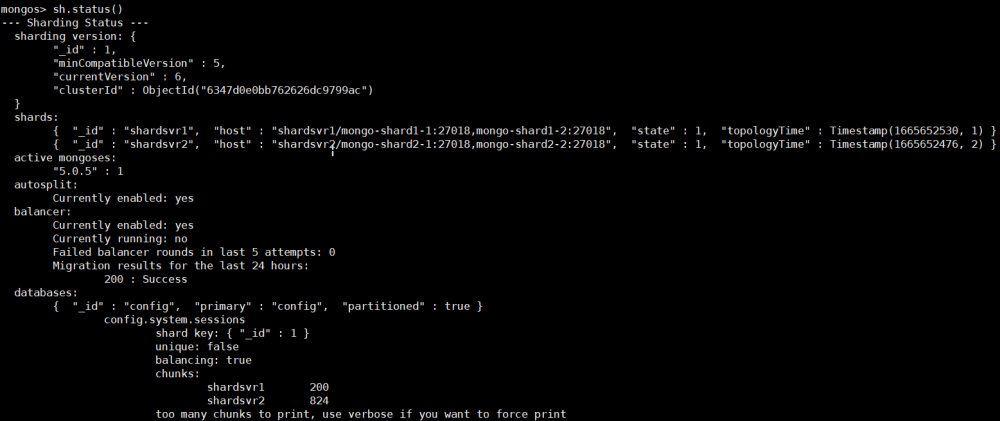
<https://www.yuque.com/caikefusiji-ydknd/exgu5x/afzwtq>

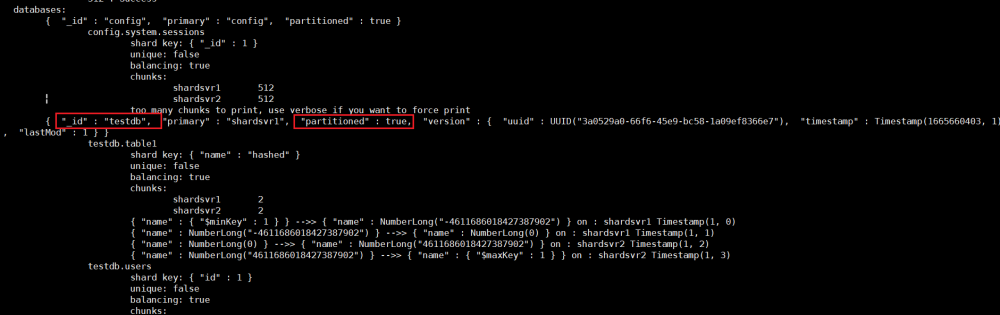
连接mongos

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

### 查看集群状态

sh.status()





"partitioned" : false } //数据库目前不支持分片（"partitioned" : false）

"partitioned" : true } //此时数据库已经支持分片

mongos> use admin  
mongos> db.runCommand({listshards:1})

### 

### 查看集群间的连接情况

mongos> db.adminCommand("connPoolStats");

mongos> db.adminCommand("connPoolStats");

{

"numClientConnections" : 0,

"numAScopedConnections" : 0,

"totalInUse" : 0,

"totalAvailable" : 3,

"totalCreated" : 6,

"totalRefreshing" : 0,

"replicaSetMatchingStrategy" : "matchPrimaryNode",

"pools" : {

"NetworkInterfaceTL-ShardRegistry" : {

"poolInUse" : 0,

"poolAvailable" : 3,

"poolCreated" : 6,

"poolRefreshing" : 0,

"mongo-config1:27019" : {

"inUse" : 0,

"available" : 1,

"created" : 4,

"refreshing" : 0

},

"mongo-config2:27019" : {

"inUse" : 0,

"available" : 1,

"created" : 1,

"refreshing" : 0

},

"mongo-config3:27019" : {

"inUse" : 0,

"available" : 1,

"created" : 1,

"refreshing" : 0

}

}

},

"hosts" : {

"mongo-config1:27019" : {

"inUse" : 0,

"available" : 1,

"created" : 4,

"refreshing" : 0

},

"mongo-config2:27019" : {

"inUse" : 0,

"available" : 1,

"created" : 1,

"refreshing" : 0

},

"mongo-config3:27019" : {

"inUse" : 0,

"available" : 1,

"created" : 1,

"refreshing" : 0

}

},

"numReplicaSetMonitorsCreated" : 3,

"replicaSets" : {

"cfgsvr" : {

"hosts" : [

{

"addr" : "mongo-config1:27019",

"ok" : true,

"ismaster" : true,

"hidden" : false,

"secondary" : false,

"pingTimeMillis" : 0.947

},

{

"addr" : "mongo-config2:27019",

"ok" : true,

"ismaster" : false,

"hidden" : false,

"secondary" : true,

"pingTimeMillis" : 0.596

},

{

"addr" : "mongo-config3:27019",

"ok" : true,

"ismaster" : false,

"hidden" : false,

"secondary" : true,

"pingTimeMillis" : 0.884

}

]

},

"shardsvr2" : {

"hosts" : [

{

"addr" : "mongo-shard2-1:27018",

"ok" : true,

"ismaster" : true,

"hidden" : false,

"secondary" : false,

"pingTimeMillis" : 0.818

},

{

"addr" : "mongo-shard2-2:27018",

"ok" : true,

"ismaster" : false,

"hidden" : false,

"secondary" : true,

"pingTimeMillis" : 0.528

},

{

"addr" : "mongo-shard2-3:27018",

"ok" : false,

"ismaster" : false,

"hidden" : true,

"secondary" : false,

"pingTimeMillis" : 0.515

}

]

},

"shardsvr1" : {

"hosts" : [

{

"addr" : "mongo-shard1-1:27018",

"ok" : true,

"ismaster" : true,

"hidden" : false,

"secondary" : false,

"pingTimeMillis" : 2.274

},

{

"addr" : "mongo-shard1-2:27018",

"ok" : true,

"ismaster" : false,

"hidden" : false,

"secondary" : true,

"pingTimeMillis" : 1.91

},

{

"addr" : "mongo-shard1-3:27018",

"ok" : false,

"ismaster" : false,

"hidden" : true,

"secondary" : false,

"pingTimeMillis" : 1.395

}

]

}

},

"ok" : 1,

"$clusterTime" : {

"clusterTime" : Timestamp(1665660089, 2),

"signature" : {

"hash" : BinData(0,"elEKP+tP7wpSwaUPbFbgplqu7LA="),

"keyId" : NumberLong("7153916193592573972")

}

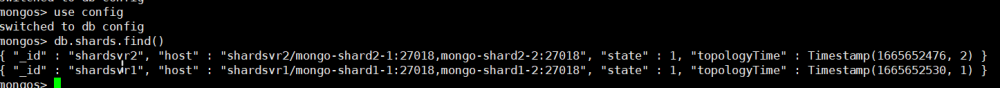
},

"operationTime" : Timestamp(1665660089, 2)

}

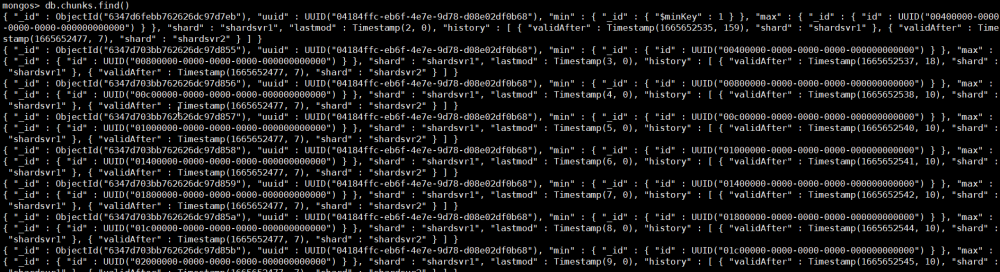
### 通过命令查看mongodb路由服务器上的shards集合会有数据展示

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



### 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()



### 测试一

#### 1 连接mongos

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

#### 2 启动分片功能

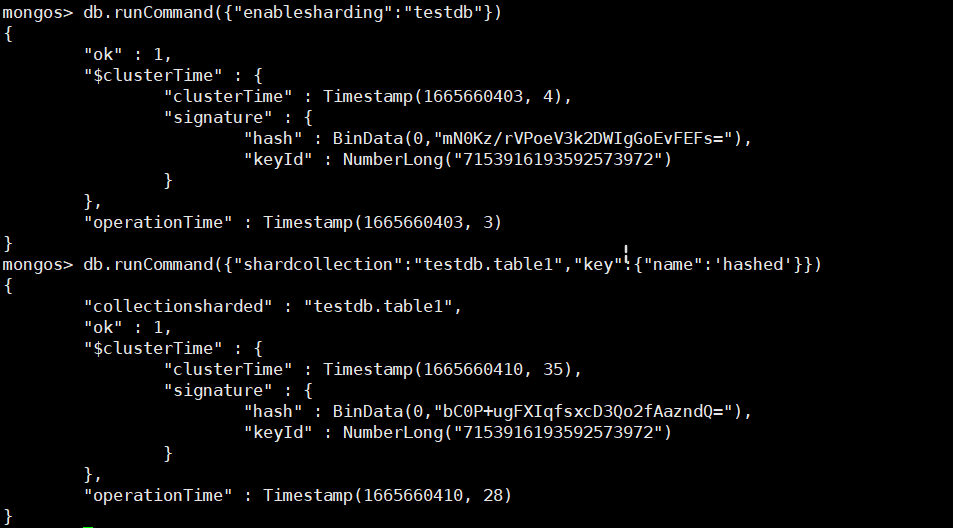
虽然数据库采用分片集群的方式部署，但如果db和collection不启用分片的话（默认是不启用的），数据不会分片存储，此时如果向集群中导入一个db，会将整个db随机存储到任意一个分片中，而不是拆分存储到多个分片。

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

use admin  
db.runCommand({"enablesharding":"testdb"})

#  设置testdb的 testcoll 表需要分片， 根据id哈希值自动分配到不同分片上

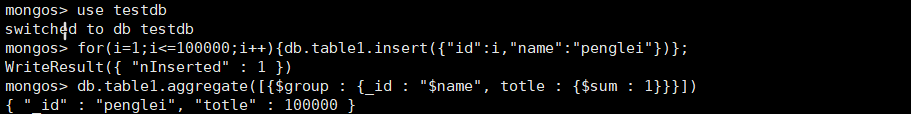
db.runCommand({"shardcollection":"testdb.table1","key":{"name":'hashed'}})



#### 4 添加数据

模拟数据1

mongos> use testdb;  
  
#插入数据  
mongos> for(i=1;i<=1000;i++){db.table1.insert({"id":i,"name":"devops"})};  
  
#总条数  
mongos> db.table1.aggregate([{$group : {\_id : "$name", totle : {$sum : 1}}}])



模拟数据2

use admin  
db.runCommand({enablesharding : "testdb" } )  
db.runCommand({shardcollection : "testdb.users",key : {id: 1} } )

var arr=[];  
for(var i=0;i<5000;i++){  
 var uid = i;  
 var name = "mongodb"+i;  
 arr.push({"id":uid,"name":name});  
}  
db.users.insertMany(arr);

#### 5 查看分片的状态

db.table1.stats();  
db.users.stats();

mongos> db.table1.stats();

{

"sharded" : true, # 是否分片

"capped" : false, #capped：集合默认值false表示不设置上限，true表示设置上限

"wiredTiger" : {

"metadata" : {

"formatVersion" : 1

},

......省略

"ns" : "testdb.table1", # namespace

"count" : 1000, # 集合总条数

"shards" : { # 分片信息

"rs\_shardsvr1" : { # id为rs\_shardsvr1的分片

"ns" : "testdb.table1",

"size" : 0,

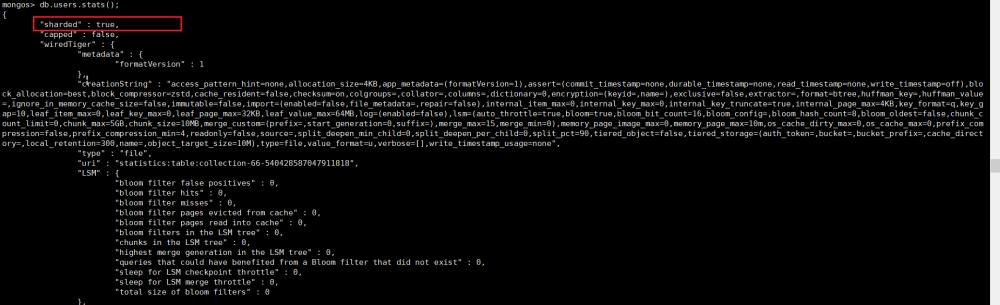
"count" : 0, # 该分片存储的条数

"storageSize" : 4096,

"freeStorageSize" : 0,

"capped" : false,

.......省略一万字.........



### 测试二

#### 1 插入数据

docker exec -it mongos\_router 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

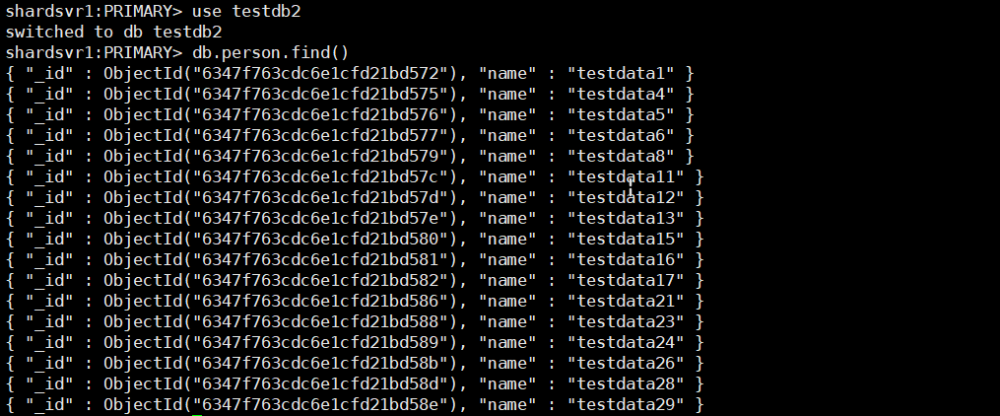
##### 1 shard1查看

**需要先创建用户**

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**

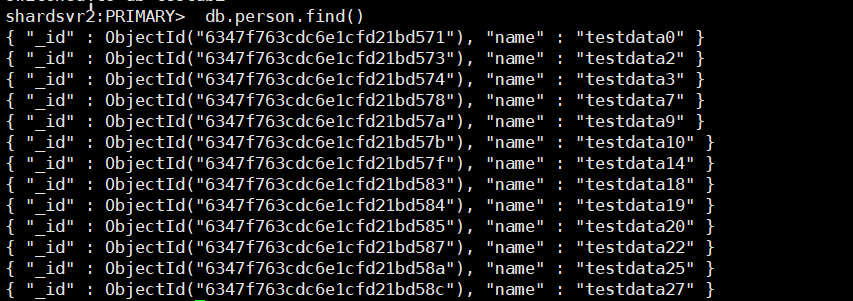
shardsvr1:PRIMARY> use testdb2  
shardsvr1:PRIMARY> db.person.find()



##### 2 shard2查看

**同shard1**

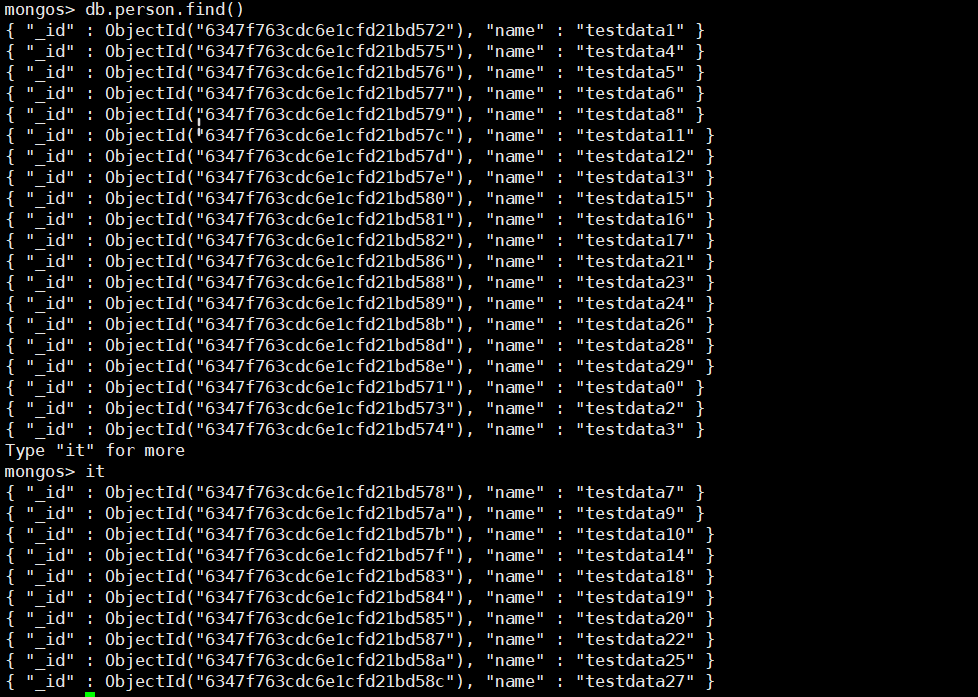
docker exec -it mongo-shard2-1 mongo 127.0.0.1:27018 -u system -p123456 --authenticationDatabase admin



##### 3 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

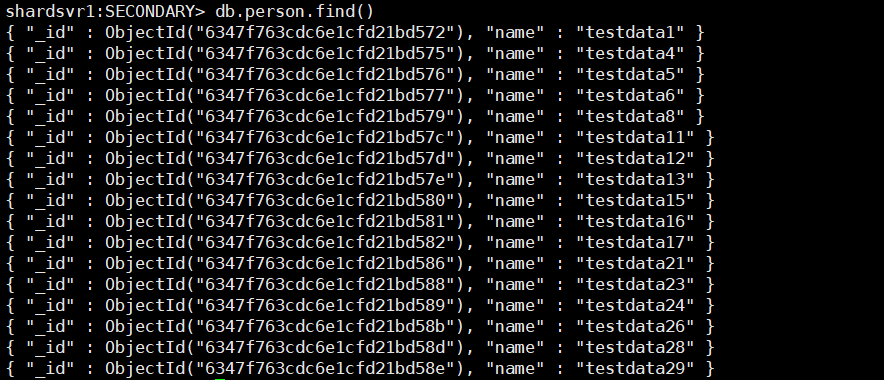


### 测试三 登录从库服务器，查看主从是否同步

#### shard1

docker exec -it mongo-shard1-2 mongo 127.0.0.1:27018 -u system -p123456 --authenticationDatabase admin

db.getMongo().setSecondaryOk();  
use testdb2  
db.person.find()

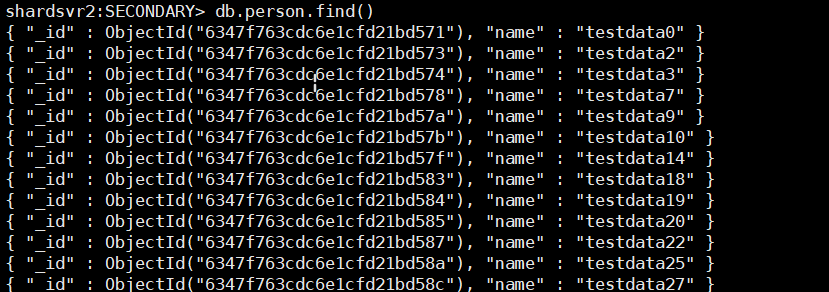


结果同主primary一致

**#db.getMongo().setSecondaryOk(); 允许SECONDARY节点查询操作。**

#### shard2

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



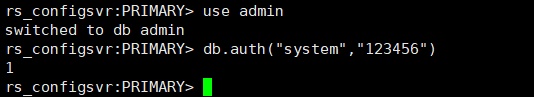
## 六 其他

### configure server 连接相关

集群创建完后 连接configure server 查询需要认证

rs\_configsvr:PRIMARY> use admin

rs\_configsvr:PRIMARY> db.auth("system","123456")



### 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.

### 报错1:

设置分片数据库、分片集合以及分片的key在admin数据库下进行，否则会报shardCollection may only be run against the admin database.错误。

## 七 参考文档

<https://blog.csdn.net/qq_38008295/article/details/110952401>

<https://www.sunjs.com/article/detail/3fa4f35574f14b86b06f1939187ea995.html>

<https://www.cnblogs.com/xingchong/p/14778146.html>

https://www.cnblogs.com/linyufeng/p/15812757.html \*

<https://blog.csdn.net/BIGmustang/article/details/108555680>

<https://cloud.tencent.com/developer/article/2028689>