Home (/fhh/fhh-doc/wikis/home) Pages (/fhh/fhh-doc/wikis/pages)
Git Access (/fhh/fhh-doc/wikis/git_access)

Mongodb test

Last edited by 任学飞 3 months ago

数据库的测试环境以及备份

目前可用服务器

```
10.90.13.159
10.90.13.160
10.90.13.161
10.90.13.162
```

规划

```
config服务器
    10.90.13.159:27100
    10.90.13.160:27100
    10.90.13.161:27100
shard1
    10.90.13.159:27101
    10.90.13.160:27101
    10.90.13.161:27101
shard2
    10.90.13.159:27102
    10.90.13.160:27102
    10.90.13.161:27102
mongos
    10.90.13.159:27103
    10.90.13.160:27103
```

start

```
如果修改了配置要重启，记得清空一下目录 dbPath(会把所有数据都清空，不要在正式环境这么做)
```

config server

```
// 在每一台机器上
sudo mkdir -p /data/logs/mongodb/config-server
sudo mkdir -p /var/run/mongodb/config-server
sudo mkdir -p /data/lib/mongo/config-server
sudo mongod --dbpath /data/lib/mongo/config-server --logpath /data/logs/mongodb/config

// 在其中一台机器上
mongo --port 27100
rs.initiate({
    _id: "config-server-test",
    configsvr: true,
    members: [
      { _id : 0, host : "10.90.13.159:27100" },
      { _id : 1, host : "10.90.13.160:27100" },
      { _id : 2, host : "10.90.13.161:27100" },
    ]
  })
```

mongos

```
// configdb后面是 配置服务器
sudo mkdir -p /data/logs/mongodb/mongos
sudo mkdir -p /var/run/mongodb/mongos
sudo mongos --logpath /data/logs/mongodb/mongos/mongos.log --fork --logappend --pidfil
```

shard 1 server

```
// 在每一台机器上
sudo mkdir -p /data/logs/mongodb/shard-1
sudo mkdir -p /var/run/mongodb/shard-1
sudo mkdir -p /data/lib/mongo/shard-1
sudo mongod --dbpath /data/lib/mongo/shard-1 --logpath /data/logs/mongodb/shard-1/mong

// 在其中一台机器上
mongo --port 27101
rs.initiate({
    _id: "shard-1-test",
    members: [
      { _id : 0, host : "10.90.13.159:27101" },
      { _id : 1, host : "10.90.13.160:27101" },
      { _id : 2, host : "10.90.13.161:27101" },
    ]
  })
```

shard 2 server

```
// 在每一台机器上
sudo mkdir -p /data/logs/mongodb/shard-2
sudo mkdir -p /var/run/mongodb/shard-2
sudo mkdir -p /data/lib/mongo/shard-2
sudo mongod --dbpath /data/lib/mongo/shard-2 --logpath /data/logs/mongodb/shard-2/mong

// 在其中一台机器上
mongo --port 27102
rs.initiate({
    _id: "shard-2-test",
    members: [
      { _id : 0, host : "10.90.13.159:27102" },
      { _id : 1, host : "10.90.13.160:27102" },
      { _id : 2, host : "10.90.13.161:27102" },
    ]
  })
```

配置分片

```
// 连接到 mongos
mongos --port 27103
// 添加replica(只需要指定repica中的一台就行)
sh.addShard( "shard-1-test/10.90.13.159:27101")
sh.addShard( "shard-2-test/10.90.13.159:27102")
```

### mongorestore --host 10.90.13.159 --port 27103 /data/backup/mongodb/20170104_1522

```
// 或者恢复到指定的db fhh_test
mongorestore --host 10.90.13.159 --port 27103 --db fhh_test /data/backup/mongodb/20170



// 链接到mongos
mongo --port 27103
> use admin
> db.runCommand({enableSharding: "fhh_test"})
> db.runCommand({shardCollection: "fhh_test.article", key: { "weMediaId" : 1, "createT
> db.runCommand({shardCollection: "fhh_test.video", key: { "weMediaId" : 1, "createTim
```

在已经运行的shard cluster上开启安全验证

创建 keyfile

```
# 这里使用keyfile作为access control是因为简单一点，而且数据库是内网的，相当是安全的
openssl rand -base64 756 > db/keyfile
chmod 400 db/keyfile
```

copy keyfile 到服务器并且设置读写权限为400

```
sudo mkdir -p /data/lib/mongo/auth
sudo cp ./db/keyfile /data/lib/mongo/auth/
sudo chmod 400 /data/lib/mongo/auth/keyfile
```

Disable the Balancer

```
mongo --port 27103
> sh.stopBalancer()
> sh.getBalancerState()
```

shutdown all mongos

```
连接到所有的mongos
# 10.90.13.159
mongo --port 27103
> db.getSiblingDB("admin").shutdownServer()
# 10.90.13.160
mongo --port 27103
> db.getSiblingDB("admin").shutdownServer()
```

shutdown all config server mongod instances

```
连接到每一个config server的mongod关闭
!!!!!!!primary最后关闭!!!!!!!!
# 10.90.13.159
mongo --port 27100
> db.getSiblingDB("admin").shutdownServer()
# 10.90.13.160
> db.getSiblingDB("admin").shutdownServer()
# 10.90.13.161
> db.getSiblingDB("admin").shutdownServer()
```

shutdown all shard replica set mongod instances

```
跟config一样，连接到所有的mongod 关闭他们
!!!!!!!primary最后关闭!!!!!!!!
# shard1
# 10.90.13.159
mongo --port 27101
> db.getSiblingDB("admin").shutdownServer()
# 10.90.13.160
> db.getSiblingDB("admin").shutdownServer()
# 10.90.13.161
> db.getSiblingDB("admin").shutdownServer()

# shard2
# 10.90.13.159
mongo --port 27102
> db.getSiblingDB("admin").shutdownServer()
# 10.90.13.160
> db.getSiblingDB("admin").shutdownServer()
# 10.90.13.161
> db.getSiblingDB("admin").shutdownServer()

# ps -ef | grep mongo 来确保都关闭了
```

重启所有的服务器，带上keyfile

config server

```
// 在每一台机器上
sudo mongod --wiredTigerCacheSizeGB 10 --dbpath /data/lib/mongo/config-server --logpat
```

shard

```
// 每台机器上
# shard1
sudo mongod --wiredTigerCacheSizeGB 10  --dbpath /data/lib/mongo/shard-1 --logpath /da
# shard2
sudo mongod --wiredTigerCacheSizeGB 10  --dbpath /data/lib/mongo/shard-2 --logpath /da
```

mongos

```
# 10.90.13.159
sudo mongos --logpath /data/logs/mongodb/mongos/mongos.log --fork --logappend --pidfil

# 10.90.13.160
sudo mongos --logpath /data/logs/mongodb/mongos/mongos.log --fork --logappend --pidfil

#现在还不能登录，登录会提示auth问题
```

设置密码 (只有一次机会，设置第一个密码后就不能再这样登录了，必须auth了)

```
在物理机器上连接mongos
# 10.90.13.159
mongo --port 27103

## 创建第一个管理员用户(只是用于赋权限的)
db.getSiblingDB("admin").createUser({
    user: "fhh_admin",

    pwd: "fhh_admin",
    roles: [ { role: "userAdminAnyDatabase", db: "admin" } ]
})
!!! 现在已经可以用这个账号登录了(只能看admin)
mongo -u "fhh_admin" -p "fhh_admin" --authenticationDatabase "admin"

## 创建其他账号
db.getSiblingDB("admin").auth("fhh_admin", "fhh_admin" )

## 超级管理员
db.getSiblingDB("admin").createUser({
    "user" : "fhh_root",
    "pwd" : "fhh_root",
    roles: [ "root" ]
})

## 创建shard管理账号，可以用来管理shard的
db.getSiblingDB("admin").createUser({
    "user" : "fhh_shard_admin",
    "pwd" : "fhh_shard_admin",
    roles: [ { "role" : "clusterAdmin", "db" : "admin" } ]
})

## 登录 shard 账号来恢复 Balancer
db.getSiblingDB("admin").auth("fhh_shard_admin", "fhh_shard_admin" )
sh.startBalancer()
```

创建平时用的用户

```
## 登录有创建权限的账号
db.getSiblingDB("admin").auth("fhh_admin", "fhh_admin" )

!!! 常用方法
db.getSiblingDB("admin").dropUser('xxx') // 删除 admin库中的xxx账号
db.getSiblingDB("admin").getUsers() // 查看admin下所有的账号

## 创建一个超级用户，拥有所有权限
db.getSiblingDB("admin").createUser({
    user: "fhh_super",
    pwd: "fhh_super",
    roles: [
          { role: "userAdminAnyDatabase", db: "admin" },
          { role: "readWriteAnyDatabase", db: "admin" },
          { role: "dbAdminAnyDatabase", db: "admin" },
          { role: "clusterAdmin", db: "admin" }
    ]
})

## fhh_test 库的读写账号
db.getSiblingDB("fhh_test").createUser({
    user: "fhh_test_rw",
    pwd: "fhh_test_rw",
    roles: [{ role: "readWrite", db: "fhh_test" }]
})

## fhh_test1 库的读写账号
db.getSiblingDB("fhh_test1").createUser({
    user: "fhh_test1_rw",
    pwd: "fhh_test1_rw",
    roles: [{ role: "readWrite", db: "fhh_test1" }]
})

## fhh_mgr_webserver_test 库的读写账号
db.getSiblingDB("fhh_mgr_webserver_test").createUser({
    user: "fhh_mgr_webserver_test_rw",
    pwd: "fhh_mgr_webserver_test_rw",
    roles: [{ role: "readWrite", db: "fhh_mgr_webserver_test" }]
})

## antiplagiarism_test 库的读写账号
db.getSiblingDB("antiplagiarism_test").createUser({
    user: "test_rw",
    pwd: "test_rw",
    roles: [{ role: "readWrite", db: "antiplagiarism_test" }]
})

## fhh_oauth2 库的读写账号
db.getSiblingDB("fhh_oauth2").createUser({
    user: "oauth_rw",
    pwd: "oauth_rw",
    roles: [{ role: "readWrite", db: "fhh_oauth2" }]
})

## backup role
db.getSiblingDB("admin").createUser({
    user: "fhh_backup",
    pwd: "fhh_backup",
    roles: ["backup"]
})

!!! 例如:
mongodump --host 10.90.13.159 --port 27103  --username fhh_backup --password fhh_backu

## restore role
db.getSiblingDB("admin").createUser({
    user: "fhh_restore",
    pwd: "fhh_restore",
    roles: ["restore"]
})

mongorestore --host 10.90.13.159 --port 27103 --username fhh_restore --password fhh_re

### 自定义role
db.getSiblingDB("fhh_test").createRole({
    role: 'fhh_product_role',
    privileges: [
        { resource: { db: "fhh_test", collection:"tmp_ac_renxf_02"}, actions: ["remove
        { resource: { db: "fhh_test", collection:""}, actions: ["convertToCapped", "cr
    ],
    roles: ["read"]
})
db.getSiblingDB("fhh_test").createUser({
    user: "fhh_product_user",
    pwd: "fhh_product_user",
    roles: ["fhh_product_role"]
})



!!! 删除role和user
db.getSiblingDB("fhh_test").dropRole('fhh_product_role')
db.getSiblingDB("fhh_test").dropUser('fhh_product_user')
```

在有账号的情况下关闭所有的服务器

```
### 需要登录超级用户

### shutdown all mongos
连接到所有的mongos
# 10.90.13.159
mongo --port 27103
> db.getSiblingDB("admin").auth("fhh_super", "fhh_super" )
> db.getSiblingDB("admin").shutdownServer()
# 10.90.13.160
mongo --port 27103

> db.getSiblingDB("admin").auth("fhh_super", "fhh_super" )
> db.getSiblingDB("admin").shutdownServer()
```

shutdown all config server mongod instances

```
连接到每一个config server的mongod关闭
!!!!!!!primary最后关闭!!!!!!!!
# 10.90.13.159
mongo --port 27100
> db.getSiblingDB("admin").auth("fhh_super", "fhh_super" )
> db.getSiblingDB("admin").shutdownServer()
# 10.90.13.160
> db.getSiblingDB("admin").auth("fhh_super", "fhh_super" )
> db.getSiblingDB("admin").shutdownServer()
# 10.90.13.161
> db.getSiblingDB("admin").auth("fhh_super", "fhh_super" )
> db.getSiblingDB("admin").shutdownServer()
```

shutdown all shard replica set mongod instances

```
跟config一样，连接到所有的mongod 关闭他们
!!!!!!!primary最后关闭!!!!!!!!
# shard1
# 10.90.13.159
mongo --port 27101
> db.getSiblingDB("admin").auth("fhh_super", "fhh_super" )
> db.getSiblingDB("admin").shutdownServer()
# 10.90.13.160
> db.getSiblingDB("admin").auth("fhh_super", "fhh_super" )
> db.getSiblingDB("admin").shutdownServer()

# 10.90.13.161
> db.getSiblingDB("admin").auth("fhh_super", "fhh_super" )
> db.getSiblingDB("admin").shutdownServer()


# shard2
# 10.90.13.159
mongo --port 27102
> db.getSiblingDB("admin").auth("fhh_super", "fhh_super" )
> db.getSiblingDB("admin").shutdownServer()
# 10.90.13.160
> db.getSiblingDB("admin").auth("fhh_super", "fhh_super" )
> db.getSiblingDB("admin").shutdownServer()
# 10.90.13.161
> db.getSiblingDB("admin").auth("fhh_super", "fhh_super" )
> db.getSiblingDB("admin").shutdownServer()

# ps -ef | grep mongo 来确保都关闭了
```

Last edited by 任学飞 3 months ago