链滴

跨区 MongoDB 的 replica 高可用验证

作者：SmiteLi

原文链接：https://ld246.com/article/1596769397500

来源网站：链滴

跨区MongoDB的replica高可用验证

背景：创建7个节点的replica：三区4个节点，四区3个节点。

- 四区不可用时（3个节点挂了），mongo集群还能正常工作
- 三区不可用时（4个节点挂了），此情况下需要人工干预

人工干预的思路为把剩下的三个节点重新配置为一个新的副本集，步骤如下：

1. 登录其中一个存活的节点，用以下命令查看和保存当前配置

```
cfg = rs.conf()

printjson(cfg)
```

2. 重新配置副本集成员：下面的4，5，6为cfg.members这个数组中的下标（下标从0开始算起），是每个节点的“_id”参数。请根据实际情况配置。

```
cfg.members = [cfg.members[5] , cfg.members[4] , cfg.members[6]]
```

3. 重新配置副本集：

```
rs.reconfig(cfg, {force : true})
```

操作后，剩下的三个存活节点会选出一个primary节点。

恢复正常

当出问题的区恢复正常后，需要把恢复正常的节点加入到新的副本集去。参考命令：

```
rs.add( "192.168.11.141:27001" )
rs.add( "192.168.11.141:27002" )
```

```
rs.add( "192.168.11.141:27003" )
rs.add( "192.168.11.141:27004" )
```

常用操作

```
use admin;
db.createUser(
  {
    user: "root",
    pwd: "root",
    roles: [ { role: "root", db: "admin" } ]
  }
)
db.auth("root","root")
show users
db.inventory.insertOne(
   { item: "canvas", qty: 100, tags: ["cotton"], size: { h: 28, w: 35.5, uom: "cm" } }
)
db.inventory.insertOne(
   { item: "canvas", qty: 100, tags: ["cotton"], size: { h: 28, w: 35.5, uom: "cm" } }
)
db.inventory.insertMany([
   { item: "journal", qty: 25, tags: ["blank", "red"], size: { h: 14, w: 21, uom: "cm" } },
   { item: "mat", qty: 85, tags: ["gray"], size: { h: 27.9, w: 35.5, uom: "cm" } },
   { item: "mousepad", qty: 25, tags: ["gel", "blue"], size: { h: 19, w: 22.85, uom: "cm" } }
])
db.inventory.find( {} )
```

- secondary的操作

```
设置 secondary 节点可读
db.getMongo().setSlaveOk()
rs.slaveOk()
```

- 添加新的数据到新的副本集：

```
db.inventory.insertMany([
   { item: "journal", qty: 26, tags: ["blank", "red"], size: { h: 14, w: 21, uom: "cm" } },
   { item: "mat", qty: 88, tags: ["gray"], size: { h: 27.9, w: 35.5, uom: "cm" } },
   { item: "mousepad", qty: 35, tags: ["gel", "blue"], size: { h: 19, w: 22.85, uom: "cm" } }
])
db.inventory.find( {} )
```

- 新副本集重新添加失效节点：

```
rs.add( "192.168.11.141:27001" )
```

失效节点加进来新副本集后数据是如何同步的?

27007是新副本集的primary，27001是失效后，再加入新副本集的节点。27001的日志如下：

```
2020-08-07T02:22:01.769+0000 I REPL     [replexec-0] Member 192.168.11.141:27007 is now i
 state PRIMARY
```

```
2020-08-07T02:22:02.303+0000 I REPL     [rsBackgroundSync] sync source candidate: 192.168.
1.141:27007
2020-08-07T02:22:02.303+0000 I ASIO     [RS] Connecting to 192.168.11.141:27007
2020-08-07T02:22:02.305+0000 I REPL     [rsBackgroundSync] Changed sync source from emp
y to 192.168.11.141:27007
```

根据同步日志来看，是全量同步的。