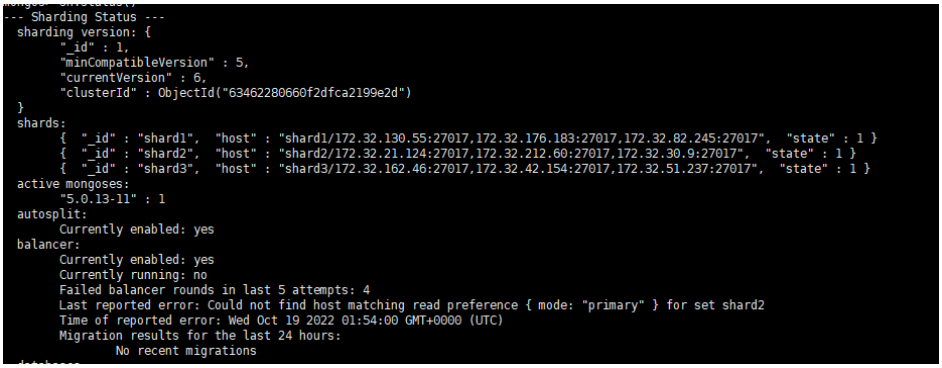
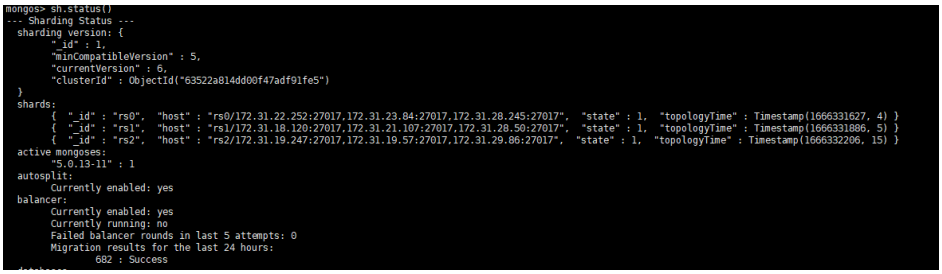
# Data sync With mongoshake ( 단방향 실시간 동기화 성공 )

현재 동기화를 진행해볼 두개의 Cluster 은 아래와 같다.




→




Source
Cluster

Target Cluster

https://github.com/alibaba/MongoShake/releases ← 최신 release 2.8.1 버전을 다운 받았다.

# tar xzvf mongo-shake-v2.6.6.tgz
명령어를 통해 압축해제 한뒤

collector.conf 를 수정해본다.
사용법이 나름 자세히 적혀있는 듯하다. ( 물론 중국어도 ^^ )

```
# if you have any problem, please visit
https://github.com/alibaba/MongoShake/wiki/FAQ
# for the detail explanation, please visit xxxx
# 如果有问题, 请先查看 FAQ 文档以及 wiki 上的说明。
# 关于各个参数的详细说明, 请参考：xxx

# current configuration version, do not modify.
# 当前配置文件的版本号, 请不要修改该值。
conf.version = 10

# --------------------------- global configuration ---------------------------
# collector name
# id 用于输出 pid 文件等信息。
id = mongoshake

# high availability option.
# enable master election if set true. only one mongoshake can become master
# and do sync, the others will wait and at most one of them become master once
# previous master die. The master information stores in the `mongoshake` db in
the source
# database by default.
# This option is useless when there is only one mongoshake running.
# 如果开启主备 mongoshake 拉取同一个源端, 此参数需要开启。
master_quorum = false

# http api interface. Users can use this api to monitor mongoshake.
# `curl 127.0.0.1:9100`.
# We also provide a restful tool named "mongoshake-stat" to
# print ack, lsn, checkpoint and qps information based on this api.
# usage: `./mongoshake-stat --port=9100`
#
全量和增量的 restful 监控端口, 可以用 curl 查看内部监控 metric 统计情况。详见 wik
i。
full_sync.http_port = 9101
incr_sync.http_port = 9100
# profiling on net/http/profile
# profiling 端口, 用于查看内部 go 堆栈。
system_profile_port = 9200

# global log level: debug, info, warning, error. lower level message will be
filter
```

```
log.level = info
# log directory. log and pid file will be stored into this file.
# if not set, default is "./logs/"
# log 和 pid 文件的目录, 如果不设置默认打到当前路径的 logs 目录。
log.dir =
# log file name.
# log 文件名。
log.file = collector.log
# log flush enable. If set false, logs may not be print when exit. If
# set true, performance will be decreased extremely
#
设置 log 刷新, false 表示包含缓存, 如果 true 那么每条 log 都会直接刷屏, 但对性能有影响;
# 反之, 退出不一定能打印所有的 log, 调试时建议配置 true。
log.flush = false

# sync mode: all/full/incr. default is incr.
# all means full synchronization + incremental synchronization.
# full means full synchronization only.
# incr means incremental synchronization only.
# 同步模式, all 表示全量+增量同步, full 表示全量同步, incr 表示增量同步。
sync_mode =all

# connect source mongodb, set username and password if enable authority.
Please note: password shouldn't contain '@'.
# split by comma(,) if use multiple instance in one replica-set. E.g.,
mongodb://username1:password1@primaryA,secondaryB,secondaryC
# split by semicolon(;) if sharding enable. E.g.,
mongodb://username1:password1@primaryA,secondaryB,secondaryC;mongodb://username
e2:password2@primaryX,secondaryY,secondaryZ
#
源 MongoDB 连接串信息, 逗号分隔同一个副本集内的结点, 分号分隔分片 sharding 实例, 免密模式
# 可以忽略“username:password@”, 注意, 密码里面不能含有'@'符号。
# 举例:
# 副本集:mongodb://username1:password1@primaryA,secondaryB,secondaryC
#
分片集:mongodb://username1:password1@primaryA,secondaryB,secondaryC;mongodb:/
/username2:password2@primaryX,secondaryY,secondaryZr
mongo_urls =
mongodb://shard_admin_user:124578@172.32.176.183:27017,172.32.130.55:27017,172
```

```
.32.82.245:27017;mongodb://shard_admin_user:124578@172.32.30.9:27017,172.32.21
.124:27017,172.32.212.60:27017;mongodb://shard_admin_user:124578@172.32.162.46
:27017,172.32.51.237:27017,172.32.42.154:27017
# please fill the source config server url if source mongodb is sharding.
mongo_cs_url
=mongodb://cluster_admin_user:124578@172.32.222.4:27017,172.32.119.244:27017,1
72.32.92.11:27017
# please give at least one mongos address if source is sharding.
# 如果源端采用 change
stream 拉取, 这里还需要配置至少一个 mongos 的地址, 多个 mongos 地址以逗号 (,) 分
割
mongo_s_url = mongodb://cluster_admin_user:124578@172.32.114.98:27018
# enable source ssl
mongo_ssl_root_ca_file =

# tunnel pipeline type. now we support rpc,file,kafka,mock,direct
# 通道模式。
tunnel = direct
# tunnel target resource url
# for rpc. this is remote receiver socket address
# for tcp. this is remote receiver socket address
# for file. this is the file path, for instance "data"
# for kafka. this is the topic and brokers address which split by comma, for
# instance: topic@brokers1,brokers2, default topic is "mongoshake"
# for mock. this is uesless
# for direct. this is target mongodb address which format is the same as
`mongo_urls`. If
# the target is sharding, this should be the mongos address.
# direct 模式用于直接写入 MongoDB, 其余模式用于一些分析, 或者远距离传输场景,
# 注意, 如果是非 direct 模式, 需要通过 receiver 进行解析, 具体参考 FAQ 文档。
# 此处配置通道的地址, 格式与 mongo_urls 对齐。
tunnel.address = mongodb://cluster_admin_user:124578@13.213.44.47:27018
# the message format in the tunnel, used when tunnel is kafka.
# "raw": batched raw data format which has good performance but encoded so
that users
# should parse it by receiver.
# "json": single oplog format by json.
# "bson": single oplog format by bson.
# 通道数据的类型, 只用于 kafka 和 file 通道类型。
# raw 是默认的类型, 其采用聚合的模式进行写入和
# 读取, 但是由于携带了一些控制信息, 所以需要专门用 receiver 进行解析。
```

```
# json 以 json 的格式写入 kafka, 便于用户直接读取。
# bson 以 bson 二进制的格式写入 kafka。
tunnel.message = json
# how many partitions will be written, use some hash function in
"incr_sync.shard_key".
#
如果目的端是 kafka, 最多启用多少个 partition, 最大不超过"incr_sync.worker"。默
认 1
tunnel.kafka.partition_number = 1
# tunnel json format, it'll only take effect in the case of tunnel.message =
json
# and tunnel == kafka. Set canonical_extended_json if you want to use
"Canonical
# Extended JSON Format", #559.
# 写入异构通道的 json 格式。如果希望使用 Canonical Extended Json
Format, 则设置为
# canonical_extended_json
tunnel.json.format =
# if tunnel == driect or kafka and enable ssl
tunnel.mongo_ssl_root_ca_file =

# connect mode:
# primary: fetch data from primary.
# secondaryPreferred: fetch data from secondary if has, otherwise
primary.(default)
# standalone: fetch data from given 1 node, no matter primary, secondary or
hidden. This is only
# support when tunnel type is direct.
#
连接模式, primary 表示从主上拉取, secondaryPreferred 表示优先从 secondary 拉取 (
默认建议值),
# standalone 表示从任意单个结点拉取。
mongo_connect_mode = secondaryPreferred

# filter db or collection namespace. at most one of these two parameters can
be given.
# if the filter.namespace.black is not empty, the given namespace will be
# filtered while others namespace passed.
# if the filter.namespace.white is not empty, the given namespace will be
# passed while others filtered.
# all the namespace will be passed if no condition given.
```

```
# db and collection connected by the dot(.).
# different namespaces are split by the semicolon(;).
# filter: filterDbName1.filterCollectionName1;filterDbName2
#
黑白名单过滤, 目前不支持正则, 白名单表示通过的 namespace, 黑名单表示过滤的 name
space,
#
不能同时指定。分号分割不同 namespace, 每个 namespace 可以是 db, 也可以是 db.coll
ection。
filter.namespace.black =
filter.namespace.white =
# some databases like "admin", "local", "mongoshake", "config", "system.views"
are
# filtered, users can enable these database based on some special needs.
# different database are split by the semicolon(;).
# e.g., admin;mongoshake.
# pay attention: collection isn't support like "admin.xxx" except
"system.views"
#
正常情况下, 不建议配置该参数, 但对于有些非常特殊的场景, 用户可以启用 admin, mo
ngoshake 等库的同步,
# 以分号分割, 例如：admin;mongoshake。
filter.pass.special.db =
# only transfer oplog commands for syncing. represent
# by oplog.op are "i","d","u".
# DDL will be transferred if disable like create index, drop databse,
# transaction in mongodb 4.0.
# 是否需要开启 DDL 同步, true 表示开启, 源是 sharding 暂时不支持开启。
# 如果目的端是 sharding, 暂时不支持 applyOps 命令, 包括事务。
filter.ddl_enable = false
# filter oplog gid if enabled.
#
如果 MongoDB 启用了 gid, 但是目的端 MongoDB 不支持 gid 导致同步会失败, 可以启用 gi
d 过滤, 将会去掉 gid 字段。
# 谨慎建议开启, shake 本身性能受损很大。
filter.oplog.gids = false

# checkpoint info, used in resuming from break point.
# checkpoint 存储信息, 用于支持断点续传。
# context.storage.url is used to mark the checkpoint store database. E.g.,
mongodb://127.0.0.1:20070
```

```
# if not set, checkpoint will be written into source mongodb(db=mongoshake)
#
checkpoint 的具体写入的 MongoDB 地址, 如果不配置, 对于副本集和分片集群都将写入
源库(db=mongoshake)
# 2.4 版本以后不需要配置为源端 cs 的地址。
checkpoint.storage.url =
# checkpoint db's name.
# checkpoint 存储的 db 的名字
checkpoint.storage.db = mongoshake
# checkpoint collection's name.
#
checkpoint 存储的表的名字, 如果启动多个 mongoshake 拉取同一个源可以修改这个表名
以防止冲突。
checkpoint.storage.collection = ckpt_default
# set if enable ssl
checkpoint.storage.url.mongo_ssl_root_ca_file =
# real checkpoint: the fetching oplog position.
# pay attention: this is UTC time which is 8 hours latter than CST time. this
# variable will only be used when checkpoint is not exist.
#
本次开始拉取的位置, 如果 checkpoint 已经存在 (位于上述存储位置) 则该参数无效,
# 如果需要强制该位置开始拉取, 需要先删除原来的 checkpoint, 详见 FAQ。
# 若 checkpoint 不存在, 且该值为 1970-01-
01T00:00:00Z, 则会拉取源端现有的所有 oplog。
# 若 checkpoint 不存在, 且该值不为 1970-01-
01T00:00:00Z, 则会先检查源端 oplog 最老的时间是否
# 大于给定的时间, 如果是则会直接报错退出。
checkpoint.start_position = 1970-01-01T00:00:00Z

# transform from source db or collection namespace to dest db or collection
namespace.
# at most one of these two parameters can be given.
# transform:
fromDbName1.fromCollectionName1:toDbName1.toCollectionName1;fromDbName2:toDbNa
me2
# 转换命名空间, 比如 a.b 同步后变成 c.d, 谨慎建议开启, 比较耗性能。
transform.namespace =

# ---------------------------- full sync configuration ----------------------
---
# the number of collection concurrence
```

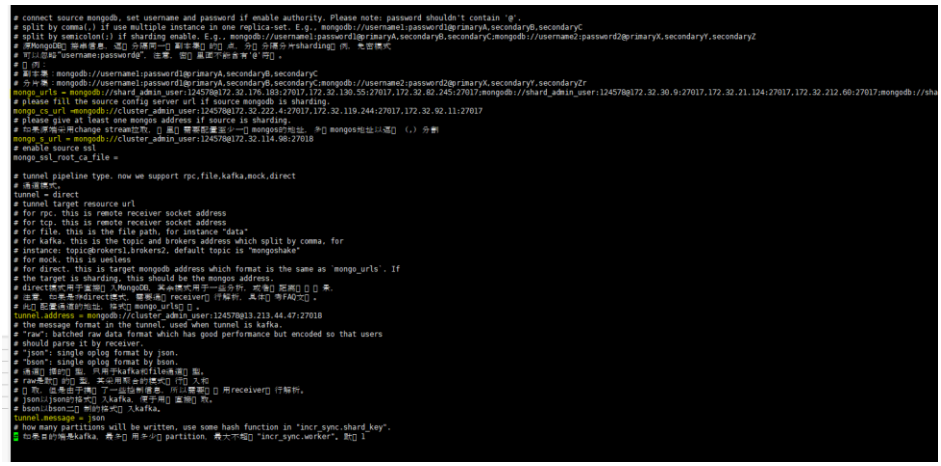
```
# 并发最大拉取的表个数，例如，6 表示同一时刻 shake 最多拉取 6 个表。
full_sync.reader.collection_parallel = 6
# the number of document writer thread in each collection.
#
同一个表内并发写的线程数，例如，8 表示对于同一个表，将会有 8 个写线程进行并发写入。
full_sync.reader.write_document_parallel = 8
# number of documents in a batch insert in a document concurrence
#
目的端写入的 batch 大小，例如，128 表示一个线程将会一次聚合 128 个文档然后再写入。
full_sync.reader.document_batch_size = 128
# max number of fetching thread per table. default is 1
# 单个表最大拉取的线程数，默认是单线程拉取。需要具备 splitVector 权限。
#
注意：对单个表来说，仅支持索引对应的 value 是同种类型，如果有不同类型请勿启用该配置项！
full_sync.reader.parallel_thread = 1
# the parallel query index if set full_sync.reader.parallel_thread. index should only has
# 1 field.
#
如果设置了 full_sync.reader.parallel_thread，还需要设置该参数，并行拉取所扫描的 index，value
#
必须是同种类型。对于副本集，建议设置_id；对于集群版，建议设置 shard_key。key 只能有 1 个 field。
full_sync.reader.parallel_index = _id

# drop the same name of collection in dest mongodb in full synchronization
#
同步时如果目的库存在，是否先删除目的库再进行同步，true 表示先删除再同步，false 表示不删除。
full_sync.collection_exist_drop = true

# create index option.
# none: do not create indexes.
# foreground: create indexes when data sync finish in full sync stage.
# background: create indexes when starting.
```

```
#
全量期间数据同步完毕后, 是否需要创建索引, none 表示不创建, foreground 表示创建前台索引,
# background 表示创建后台索引。
full_sync.create_index = none

# convert insert to update when duplicate key found
# 如果_id 存在在目的库, 是否将 insert 语句修改为 update 语句。
full_sync.executor.insert_on_dup_update = false
# filter orphan document for source type is sharding.
# 源端是 sharding, 是否需要过滤 orphan 文档
full_sync.executor.filter.orphan_document = false
# enable majority write in full sync.
# the performance will degrade if enable.
# 全量阶段写入端是否启用 majority write
full_sync.executor.majority_enable = false

# ---------------------------- incrmental sync configuration ------------------
---------
# fetch method:
# oplog: fetch oplog from source mongodb (default)
# change_stream: use change to receive change event from source mongodb,
support MongoDB >= 4.0.
# we recommand to use change_stream if possible.
incr_sync.mongo_fetch_method = oplog

# After the document is updated, the fields that only need to be updated are
set to false,
# and the contents of all documents are set to true
# 更新文档后,只需要更新的字段则设为 false,需要全部文档内容则设为 true
# 只在 mongo_fetch_method = change_stream 模式下生效, 且性能有所下降
incr_sync.change_stream.watch_full_document = false

# global id. used in active-active replication.
# this parameter is not supported on current open-source version.
#
gid 用于双活防止环形复制, 目前只用于阿里云云上 MongoDB, 如果是阿里云云上实例互相同步
# 希望开启 gid, 请联系阿里云售后, sharding 的有多个 gid 请以分号(;)分隔。
incr_sync.oplog.gids =
```

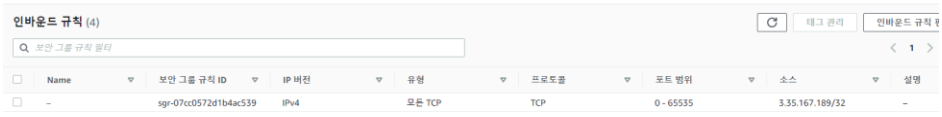
```
# distribute data to different worker by hash key to run in parallel.
# [auto]              decide by if there has unique index in collections.
#                             use `collection` if has unique index otherwise
use `id`.
# [id]                shard by ObjectId. handle oplogs in sequence by unique
_id
# [collection] shard by ns. handle oplogs in sequence by unique ns
#
hash 的方式, id 表示按文档 hash, collection 表示按表 hash, auto 表示自动选择 hash
类型。
# 如果没有索引建议选择 id 达到非常高的同步性能, 反之请选择 collection。
incr_sync.shard_key = collection
# if shard_key is collection, and users want to improve performance when some
collections
# do not have unique key.
#
对于按 collection 哈希, 如果某些表不具有唯一索引, 则可以设置按_id 哈希以提高并
发度。
#
用户需要确认该表不会创建唯一索引, 一旦检测发现存在唯一索引, 则会立刻 crash 退出
。
# 例如, db1.collection1;db2.collection2, 不支持仅指定 db
incr_sync.shard_by_object_id_whitelist =

# oplog transmit worker concurrent
# if the source is sharding, worker number must equal to shard numbers.
# 内部发送 (写目的 DB) 的 worker 数目, 如果机器性能足够, 可以提高 worker 个数。
incr_sync.worker = 8

# how many writing threads will be used in one worker.
# 对于目的端是 kafka 等非 direct
tunnel, 启用多少个序列化线程, 必须为"incr_sync.worker"的倍数。
# 默认为"incr_sync.worker"的值。
incr_sync.tunnel.write_thread = 8

# set the sync delay just like mongodb secondary slaveDelay parameter. unit
second.
#
设置目的端的延迟, 比如延迟源端 20 分钟, 类似 MongoDB 本身主从同步 slaveDelay 参数
, 单位：秒
# 0 表示不启用
```

```
incr_sync.target_delay = 0

# memory queue configuration, plz visit FAQ document to see more details.
# do not modify these variables if the performance and resource usage can
# meet your needs.
# 内部队列的配置参数, 如果目前性能足够不建议修改, 详细信息参考 FAQ。
# batch_queue_size：每个 worker 线程的队列长度, worker 线程从此队列取任务
# batching_max_size：一次分发给 worker 的任务最多包含多少个文档
#
buffer_capacity：PendingQueue 队列中一个 buffer 至少包含的文档个数, 进行序列化
incr_sync.worker.batch_queue_size = 64
incr_sync.adaptive.batching_max_size = 1024
incr_sync.fetcher.buffer_capacity = 256

# --- direct tunnel only begin ---
# if tunnel type is direct, all the below variable should be set
# 下列参数仅用于 tunnel 为 direct 的情况。

# oplog changes to Insert while Update found non-exist (_id or unique-index)
# 如果_id 不存在在目的库, 是否将 update 语句修改为 insert 语句。
incr_sync.executor.upsert = false
# oplog changes to Update while Insert found duplicated key (_id or unique-
index)
# 如果_id 存在在目的库, 是否将 insert 语句修改为 update 语句。
incr_sync.executor.insert_on_dup_update = false
# db. write duplicated logs to mongoshake_conflict
# 如果写入存在冲突, 记录冲突的文档。选项：db, none
# db：冲突写到目的 DB 的 mongshake_conflict 库中
incr_sync.conflict_write_to = none

# enable majority write in incrmental sync.
# the performance will degrade if enable.
# 增量阶段写入端是否启用 majority write
incr_sync.executor.majority_enable = false

# --- direct tunnel only end ---

# 特殊字段, 标识源端类型, 默认为空。阿里云 MongoDB
serverless 集群请配置 aliyun_serverless
special.source.db.flag =
```

로그의 기본 위치, source MongoDB URL , Tunnel MongoDB URI 등을 입력해주자.



아무래도 다른 region 간의 통신이 필요하다보니 tunnel mongos address 는 public address 를 주고

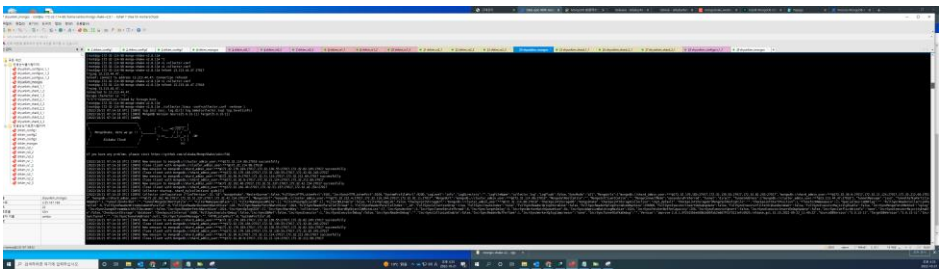



( Port 를 어떤것을 쓸지를 모르겠어서 일단 모든 Port 를 열었는데 conf 파일을 뒤져보니 9101 9100 9200 번을 사용하는 듯하다. )

혹시나 하는 마음으로 telnet 을 이용하여 각 mongos 끼리 통신이 되고 있는지 확인한 후

./collector.linux -conf=collector.conf -verbose 1

명령어로 실행해 보았다

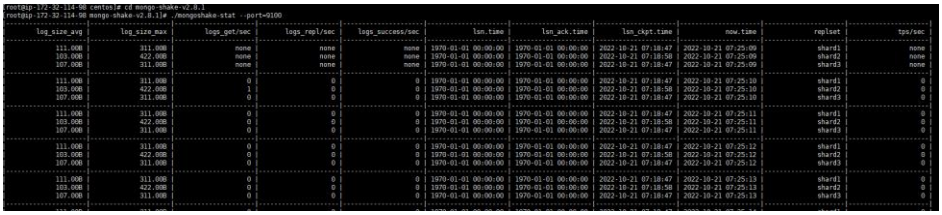


이제 Target mongos 에서 동기화가 진행 되었는지 확인해 보았다

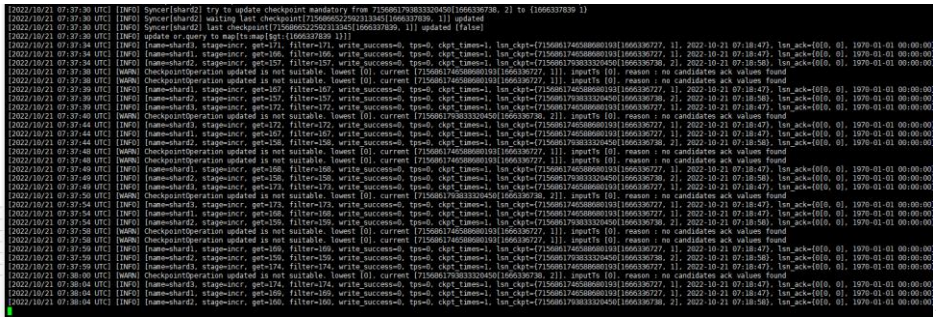
```
mongos> db.users.getShardDistribution()

Shard rs2 at rs2/172.31.19.247:27017,172.31.19.57:27017,172.31.29.86:27017
 data : 33.07MiB docs : 333819 chunks : 2
 estimated data per chunk : 16.53MiB
 estimated docs per chunk : 166909

Shard rs1 at rs1/172.31.18.120:27017,172.31.21.107:27017,172.31.28.50:27017
 data : 32.91MiB docs : 332230 chunks : 2
 estimated data per chunk : 16.45MiB
 estimated docs per chunk : 166115

Shard rs0 at rs0/172.31.22.252:27017,172.31.23.84:27017,172.31.28.245:27017
 data : 33.08MiB docs : 333951 chunks : 2
 estimated data per chunk : 16.54MiB
 estimated docs per chunk : 166975

Totals
 data : 99.07MiB docs : 1000000 chunks : 6
 Shard rs2 contains 33.38% data, 33.38% docs in cluster, avg obj size on shard : 103B
 Shard rs1 contains 33.22% data, 33.22% docs in cluster, avg obj size on shard : 103B
 Shard rs0 contains 33.39% data, 33.39% docs in cluster, avg obj size on shard : 103B

mongos>
```

와우.....

성공인듯하다. (로그)



현재 1 초에 한번 계속 동기화를 진행하고 있는듯하다



후보가 없다고 하는 걸로 보아 새로운 데이터를 넣어주면 실시간 동기화를 해주지 않을 까?

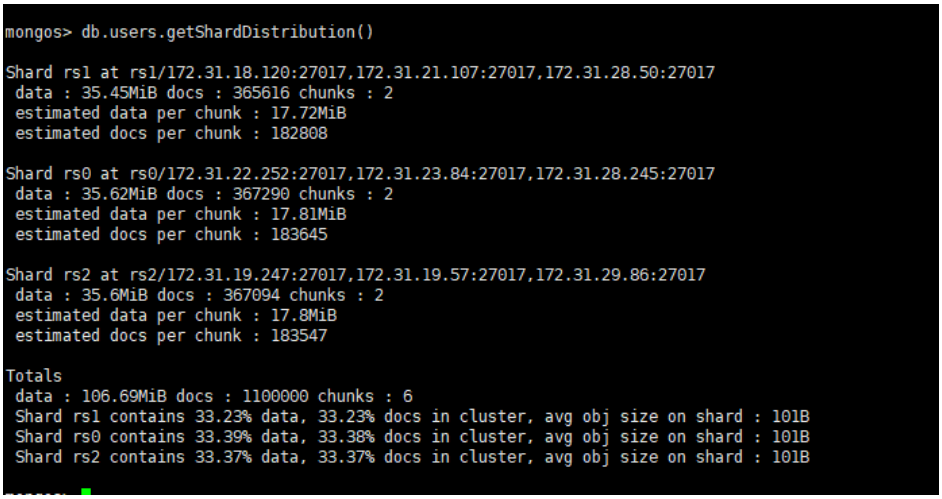
십만개의 데이터를 Source mongos 에 넣어 보겠다.

```
mongos> use test
switched to db test
mongos> for (i=0;i<100000;i++){
... db.users.insertOne(
... {
... "i":i+1000000,
... "username":"user"+i+1000000,
... "height":Math.floor(Math.random()*41 +160)
... }
... );
... }
```

입력과 동시에 mongoshake 가 일을 하는 듯하다

```
mongos> db.users.getShardDistribution()

Shard rs0 at rs0/172.31.22.252:27017,172.31.23.84:27017,172.31.28.245:27017
 data : 33.6MiB docs : 340804 chunks : 2
 estimated data per chunk : 16.8MiB
 estimated docs per chunk : 170402

Shard rs2 at rs2/172.31.19.247:27017,172.31.19.57:27017,172.31.29.86:27017
 data : 33.59MiB docs : 340693 chunks : 2
 estimated data per chunk : 16.79MiB
 estimated docs per chunk : 170346

Shard rs1 at rs1/172.31.18.120:27017,172.31.21.107:27017,172.31.28.50:27017
 data : 33.42MiB docs : 338946 chunks : 2
 estimated data per chunk : 16.71MiB
 estimated docs per chunk : 169473

Totals
 data : 100.62MiB docs : 1020443 chunks : 6
 Shard rs0 contains 33.39% data, 33.39% docs in cluster, avg obj size on shard : 103B
 Shard rs2 contains 33.38% data, 33.38% docs in cluster, avg obj size on shard : 103B
 Shard rs1 contains 33.21% data, 33.21% docs in cluster, avg obj size on shard : 103B

mongos> db.users.getShardDistribution()

Shard rs0 at rs0/172.31.22.252:27017,172.31.23.84:27017,172.31.28.245:27017
 data : 33.62MiB docs : 341071 chunks : 2
 estimated data per chunk : 16.81MiB
 estimated docs per chunk : 170535

Shard rs1 at rs1/172.31.18.120:27017,172.31.21.107:27017,172.31.28.50:27017
 data : 33.44MiB docs : 339201 chunks : 2
 estimated data per chunk : 16.72MiB
 estimated docs per chunk : 169600

Shard rs2 at rs2/172.31.19.247:27017,172.31.19.57:27017,172.31.29.86:27017
 data : 33.61MiB docs : 340938 chunks : 2
 estimated data per chunk : 16.8MiB
 estimated docs per chunk : 170469

Totals
 data : 100.68MiB docs : 1021210 chunks : 6
 Shard rs0 contains 33.39% data, 33.39% docs in cluster, avg obj size on shard : 103B
 Shard rs1 contains 33.21% data, 33.21% docs in cluster, avg obj size on shard : 103B
 Shard rs2 contains 33.38% data, 33.38% docs in cluster, avg obj size on shard : 103B
```

실시간으로 늘어나고 있다;;




현재 단방향 동기화가 정상적으로 작동하고 있는 것으로 보인다.