# Giới thiệu

# Khởi tạo replica set

(đơn giản)

<https://docs.mongodb.com/manual/tutorial/deploy-replica-set/>

mongodb version: 5.0

ubuntu server 18.04 LTS

## B1: install mongo

## B2: update config

(VD: /etc/mongod.conf )

cập nhật replication.replSetName

VD:

replication:

replSetName: rs0

CAUTIONS: repSetName của config các node phải giống nhau

(nếu cần) security

khi bật bảo mật cần bật 2 config security.authorization và security.keyFile

VD:

security:

authorization: enabled

keyFile: /home/tuana9a/key0

CAUTIONS:

set quyền access keyFile VD: -rw------- 1 mongodb mongodb 65 Aug 5 04:42 key0

keyFile có thể chứa nhiều key

độ dài key từ 6 đến 1024 ký tự chỉ chứa các ký tự nằm trong base64

các members phải share ít nhất chung một key

keyFile details:

<https://docs.mongodb.com/manual/tutorial/enforce-keyfile-access-control-in-existing-replica-set/>

<https://docs.mongodb.com/manual/core/security-internal-authentication/#std-label-internal-auth-keyfile>

## B3: start or restart instance

với instance mới chỉ cần start là chạy

với instance đang chạy thì cần restart

## B4: initiate replica set

cách 1: khởi tạo tất cả cùng lúc

- kết nối tới 1 node bất kì

- khởi tạo

rs.initiate({

\_id : "rs0", //giống với replication.repSetName trong /etc/mongod.conf

members: [

{ \_id: 0, host: "vm-01.local:27017" },

{ \_id: 1, host: "vm-02.local:27017", priority: 0 },

{ \_id: 2, host: "vm-03.local:27017", arbiterOnly: true }, // arbiter

{ \_id: 3, host: "vm-04.local:27017", priority:0, hidden: true }, // hidden

{ \_id: 4, host: "vm-05.local:27017", priority:0, slaveDelay: 1800 }, // delay 30 min

{ \_id: 5, host: "vm-06.local:27017", priority:0, votes:0 }, // non votes

]

})

TIPS: sử dụng DNS hostname thay IP sẽ tránh việc reconfig toàn bộ replica set khi IP thay đổi

cách 2: khởi tạo primary trước sau đó thêm lần lượt các members

- kết nối tới primary

- khởi tạo replica set tại primary

rs.initiate()

- thêm lần lượt các members sau đó [xem ở đây](#_6wos6ibr9ulx)

rs.add( { host: "vm-02.local:27017", priority: 0, votes: 0 } )

## B5: check success

VD:

rs.config().members

hoặc rs.status().members

{

"set" : "rs0",

...

"members" : [

{

"name" : "vm-01.local:27017"

...

},

{

"name" : "vm-02.local:27017"

...

},

{

"name" : "vm-03.local:27017",

...

}

],

"ok" : 1,

...

}

## B6: update dns

## B7: (bonus) update dns

cập nhật dns với bản ghi srv để support phân giải toàn độ host của các node từ một host duy nhất

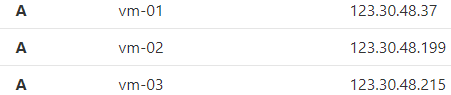
mongodb có hỗ trợ dns loại srv

hỗ trợ phía client chỉ cần một địa chỉ sẽ phân giải ra các hostname của các node trong replica set

hostname của các node trong replica set

domain gốc: tuana9a.tech

vm-01.tuana9a.tech, vm-02.tuana9a.tech, vm-03.tuana9a.tech

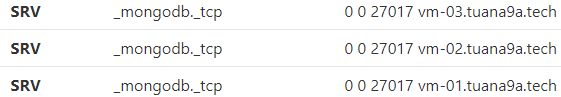


thêm các bản ghi loại srv trỏ tới các node trong replicaset rs

service = \_mongodb // bắt buộc

name = rs // rs.tuana9a.tech sẽ là địa chỉ connect, có thể đặt tên bất kì

VD: name = rs0.iloveyou sẽ cần connect tới rs0.iloveyou.tuana9a.tech



CAUTIONS: các bản ghi srv cùng domain với các node

sau đó phía client connect như sau

mongodb+srv://rs.tuana9a.tech

# Replica set members

## Primary

<https://docs.mongodb.com/manual/core/replica-set-primary/>

- The primary is the only member that receives write operations

- MongoDB applies write operations on the primary and then records it on the primary's oplog

+ secondary members replicate primary's oplog and apply the operations to their data sets.

- By default, an application directs its read operations to the primary member

- The replica set can have at most one primary

- If primary becomes unavailable, an election determines the new primary <https://docs.mongodb.com/manual/core/replica-set-elections/>

## Arbiter

- do not hold data.

- They can, however, participate in elections for a new primary.

- An arbiter has exactly 1 election vote.

- In version 3.6 arbiter priority is 0 (before the default is 1)

CAUTIONS: Do not run an arbiter on systems that also host the primary or the secondary members of the replica set.

## Secondary

<https://docs.mongodb.com/manual/core/replica-set-secondary/>

- A secondary maintains a copy of the primary's data set

+ it applies operations direct from the primary's oplog to its own data set in an asynchronous process

- Clients cannot write data to secondaries, clients can read data from secondary

<https://docs.mongodb.com/manual/core/read-preference/>

- You can configure a secondary to:

+ Priority 0 Member: Prevent it from becoming a primary in an election, and reside in a secondary data center or to serve as a cold standby.

+ Hidden Member: Prevent applications from reading from it, which allows it to run applications that require separation from normal traffic.

+ Delayed Member: Keep a running "historical" snapshot for use in recovery from certain errors, such as unintentionally deleted databases

### Priority 0

<https://docs.mongodb.com/manual/core/replica-set-priority-0-member/>

+ cannot become primary and cannot trigger elections.

+ can acknowledge write operations issued with write concern of w : <number>

+ For "majority" write concern, the priority 0 member must also be a voting member (votes > 0) to acknowledge the write

<https://docs.mongodb.com/manual/reference/write-concern/#std-label-write-concern>

+ maintain a copy of the data set, accept read operations, and vote in elections

### Hidden

<https://docs.mongodb.com/manual/core/replica-set-hidden-member/>

+ maintains a copy of the primary's data set but is invisible to client applications

+ must always be priority 0 members and so cannot become primary

+ Clients will not distribute reads with the appropriate read preference to hidden members

+ receive no traffic other than basic replication

+ Hidden replica set members can acknowledge write operations issued with w: <number>.

+ must also be voting members (votes > 0) to acknowledge the "majority" write operation.

+ Non-voting (votes = 0) cannot contribute to acknowledging write operations with majority write concern.

### Delay

<https://docs.mongodb.com/manual/core/replica-set-delayed-member/>

+ contains copies of data but earlier (or delay, previous state)

VD: current time is 09:52 and a member has a delay of an hour, the delayed member has no operation more recent than 08:52.

+ it is possible to recover from dropped databases and collections.

+ Must be priority 0 members. Set the priority to 0 to prevent a delayed member from becoming primary.

+ Should be hidden members. Always prevent applications from seeing and querying delayed members.

+ Do vote in elections for primary, if members[n].votes is set to 1.

|  | priority | votes | hidden | slaveDelay | write\_concern | read\_reference |
| --- | --- | --- | --- | --- | --- | --- |
| priority 0 | 0 |  | False |  | <number> |  |
| priority 0 | 0 | >0 | False |  | <majority> |  |
| hidden | 0 |  | True |  | <number> | can't read |
| hidden | 0 | >0 | True |  | <majority> | can't read |
| delay | 0 |  | True | >0 |  |  |

# Quản lý members

## Add member

B1: install mongo

B2: update files config

cập nhật replication.replSetName

replication:

replSetName: rs0

(nếu cần) enable authentication, add keyFile chung của Replica set

lấy keyFile từ replica set hiện tại

cập nhật security.keyFile trong config trỏ tới keyFile đã lấy

security:

authorization: enabled

keyFile: /home/tuana9a/key0

B3: restart instance cập nhật config (nếu cần)

B4: kết nối tới primary

B5: add member

rs.add({host: "vm-03.local:27017"}) // add secondary member

rs.addArb("vm-05.local:27017") // add arbiter member

B4: kiểm tra kết quả

rs.config().members

hoặc rs.status().members

- thuộc tính của member: <https://docs.mongodb.com/manual/reference/replica-configuration/#mongodb-rsconf-rsconf.members>

## Update - Replace member

B1: kết nối tới replica set

B2: update config

lấy config

cfg = rs.conf()

cập nhật config member tương ứng

cfg.members[0].host = "mongo2.example.net" // replace a node

cfg.members[1].priority=10 // change primary priority

reconfig replica set

rs.reconfig(cfg)

B3: kiểm tra kết quả

rs.config().members

hoặc rs.status().members

## Remove member

B1: kết nối tới replica set

B2: remove member

rs.remove("vm-03.local:27017")

B3: kiểm tra kết quả

rs.config().members

hoặc rs.status().members

# Automatic Failover

## Elections

<https://docs.mongodb.com/manual/replication/#std-label-replication-auto-failover>

<https://docs.mongodb.com/manual/core/replica-set-elections/>

Heartbeats

- Replica set members send heartbeats (pings) to each other (including primary) every two seconds.

- If a heartbeat does not return within 10 seconds, other members mark the delinquent member as inaccessible.

Member Priority

- thuật toán chọn primary sẽ "cố gắng" chọn member có priority cao nhất trong config sẽ request election tới các node khác

- member có priority cao hơn mà request election sớm hơn thì tỉ lệ thành primary cao hớn

- tuy nhiên có thể member có priority thấp hơn trở thành primary tạm thời (kể cả tồn tại member khác có priority cao hơn)

nhưng các members sẽ tiếp tục request election cho đến lúc primary là member có priority cao nhất

- member priority = 0 không thể làm primary và không "trigger" election, details

<https://docs.mongodb.com/manual/core/replica-set-priority-0-member/>

Voting Member

<https://docs.mongodb.com/manual/core/replica-set-elections/#voting-members>

- All replica set members that have their members[n].votes setting equal to 1 vote in elections

+ member có votes = 0 không tham gia vote (chính là Non Voting Member)

+ member có priority lớn hơn 0 không thể set vote = 0

+ member có votes = 0 phải set priority = 0

- Only voting members in the following states are eligible to vote:

PRIMARY

SECONDARY

STARTUP2 (unless the member was newly added to the replica set)

RECOVERING

ARBITER

ROLLBACK

Non-Voting Members

- non-voting members do not vote in elections

- hold copies of the replica set's data and can accept read operations from client applications.

- Non-voting members cannot contribute to acknowledging write operations with "majority" write concern.

- Because a replica set can have up to 50 members, but only 7 voting members

+ non-voting members allow a replica set to have more than seven members.

- Non-voting (i.e. votes is 0) members must have priority of 0.

# Data Synchronization

<https://docs.mongodb.com/v4.4/core/replica-set-sync/>

## OpLog

- là record đặc biệt nằm ở collection local.oplog.rs ghi lại toàn bộ operations on data

- một bản ghi oplog chứa các thông tin

op: loại của operation

i insert

u update

d delete

c ? command

n ? no op

ts: timestamp

(64bit) bson timestamp = time\_t + ordinal

(32bit) time\_t số giây từ Unix Epoch

(32bit) ordinal tăng trong 1s

prevOpTime: link với op trước đó

o: object liên quan tới op đó

op: "i" thì o là dữ liệu được insert

ui: ?

wall: ?

lsid: ? server sessions

txnNumber: ? server sessions

- nếu ghi đạng batch op log vẫn xé lẻ thành các oplog con

+ các oplog con này có link với nhau = prevOpTime(ts, t)

- oplog "no op"

<https://github.com/mongodb/specifications/blob/master/source/max-staleness/max-staleness.rst#primary-must-write-periodic-no-ops>

kịch bản:

lần cuối write operation cách 1 tiếng

client kết nối và yêu cầu đọc data và cùng lúc đó có write operation vào primary

do vậy ở primary thì lần cuối write cách vài s, op này chưa được replicate sang các secondary

dẫn tới client detect secondary lần cuối ghi cách 1 tiếng primary

client tưởng secondary bị "stale" khiên một lượng lớn request điều hướng lại về primary

dẫn tới spike read ở primary

bản ghi tự động "no op" này để giúp client không nhận nhầm rằng secondary bị stale

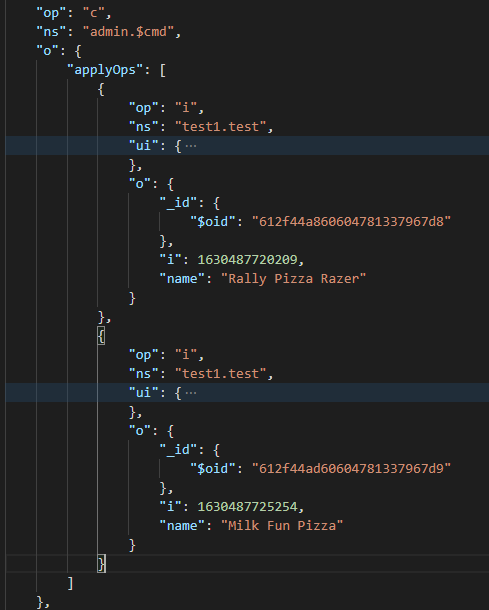
khi liên tục gửi op log "no op" để cập nhật lastWrite của secondary

- transaction

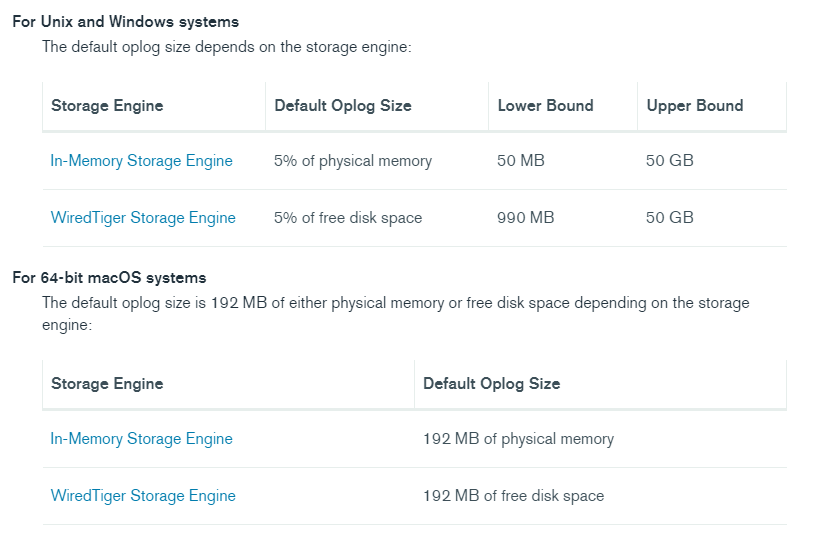
<https://docs.mongodb.com/v4.4/reference/command/applyOps/>

với transaction mongodb sẽ sử dụng applyOps (là một internal command sử dụng bởi instance)

và toàn bộ op trong một transaction sẽ được gom làm 1 oplog applyOps lên các secondary khác



- OpLog size



chặn dưới là 1GB

## Initial Sync

- Initial sync copies all the data from one member of the replica set to another member.

- See Initial Sync Source Selection for more information on initial sync source selection criteria.

<https://docs.mongodb.com/manual/core/replica-set-sync/#std-label-replica-set-initial-sync-source-selection>

Process:

- Clones all databases except local, mongod scans every collection in each source database and inserts all data into its own database

+ Changed in 3.4: Initial sync builds all collection indexes as the documents are copied for each collection.

In earlier versions, only the \_id indexes are built during this stage.

+ Changed in 3.4: Initial sync pulls newly added oplog records during the data copy.

The local database is used to temporarily store these oplog for the duration of data copy stage.

- Using the oplog from the source, the mongod updates its data set to reflect the current state of the replica set.

- When the initial sync finishes, the member transitions from STARTUP2 to SECONDARY.

## Sync Error

- nếu trong quá trình sync gặp vấn đề không phải tạm thời (network shut down) thì sẽ tự động sync lại từ đầu

+ từ phiên bản 4.4, khi sync gặp vấn đề tạm thời (mạng chập chờn, drop collection, collection rename, ...)

thì có thể cố gắng tiếp tục quá trình sync với điều kiện sync source cũng phải phiên bản hơn 4.4, nếu phiên

bản cũ hơn thì sẽ sync lại từ đầu

- By default, the secondary tries to resume initial sync for 24 hours.

+ version 4.4 adds the initialSyncTransientErrorRetryPeriodSeconds parameter

for controlling the amount of time to attempt to resume initial sync.

+ If the secondary cannot successfully resume the initial sync process during the configured time period,

it selects a new healthy source from the replica set and restarts the initial sync from the beginning.

- The secondary attempts to restart the initial sync up to 10 times before returning a fatal error.

## Runtime Sync - Replication

Secondary sẽ liên tục "replicate" dữ liệu sau initial sync bằng cách

copy oplog từ "sync source" và ốp các operation này vào database theo quy trình async

Secondary trong lúc sync có thể dựa vào các metrics, status, ping time, trạng thái, ... để tính độ trễ, độ stale, ... của "sync source"

từ đó có thể tự động chuyển sang "sync source" có các metrics tốt hơn

việc chọn "sync source" chi tiết phía dưới

<https://docs.mongodb.com/manual/core/replica-set-sync/#replication-sync-source-selection>

Streaming Replication

- Starting in MongoDB 4.4, sync from sources send a continuous stream of oplog entries to their syncing secondaries.

- Streaming replication mitigates replication lag in high-load and high-latency networks. It also:

Reduces staleness for reads from secondaries.

Reduces risk of losing write operations with w: 1 due to primary failover.

Reduces latency on write operations with w: "majority" and w: >1 (that is, any write concern that requires waiting for replication).

- Prior to 4.4, secondaries fetched batches of oplog entries by issuing a request to their sync from source and waiting for a response.

- This required a network roundtrip for each batch of oplog entries.

- version 4.4 adds the oplogFetcherUsesExhaust startup parameter for disabling streaming replication and using the older replication behavior.

- Set the oplogFetcherUsesExhaust parameter to false only if there are any resource constraints on the sync from source

or if you wish to limit MongoDB's usage of network bandwidth for replication.

Multithreaded Replication

- MongoDB applies write operations in batches using multiple threads to improve concurrency.

- MongoDB groups batches by document id (WiredTiger) and simultaneously applies each group of operations using a different thread.

- MongoDB always applies write operations to a given document in their original write order.

Changed in version 4.0.

- Starting in MongoDB 4.0, read operations that target secondaries and are configured with a read concern level of "local" or "majority"

will now read from a WiredTiger snapshot of the data if the read takes place on a secondary where replication batches are being applied.

- Reading from a snapshot guarantees a consistent view of the data,

and allows the read to occur simultaneously with the ongoing replication without the need for a lock.

- As a result, secondary reads requiring these read concern levels no longer need to wait for replication batches to be applied,

and can be handled as they are received.

## Resync a Member

<https://docs.mongodb.com/manual/tutorial/resync-replica-set-member/>

một tình huống hi hữu khi runtime vì một lí do nào đó secondary không thể bắt kịp với primary

network partition, mất mạng, mạng chậm và bản ghi oplog mới nhất của secondary cũ hơn oplog cũ nhất của "sync source"

tức những oplog cho việc tiếp tục sync đã bị xoá

member rơi vào trạng thái stale vĩnh viễn

giải pháp là sync lại từ đầu member này

## Rollback during Failover

<https://docs.mongodb.com/manual/core/replica-set-rollbacks/>

- xảy ra khi primary die (hoặc can not reach) dùng với write operation chưa được replicate sang các secondary

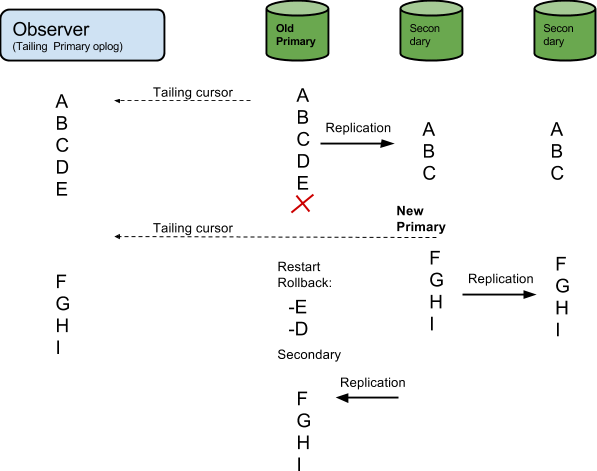
dẫn tới khi rejoin replica set gây không nhất quán dữ liệu với các secondaries trước đó

dẫn tới primary revert write operation (revert) để đảm bảo dữ liệu nhất quán

- việc rollback chỉ cần thiết khi write chưa được replica tới các secondaries khi primary die

điều này "khá" khó xảy ra, nguyên nhân đa số do network partition khi secondary không bắt

kịp với primary (lượng write operation lớn) nên nếu xảy ra sẽ dẫn tới lượng rollback là rất lớn



set default write concern majority + journal

<https://docs.mongodb.com/v4.4/reference/replica-configuration/#mongodb-rsconf-rsconf.writeConcernMajorityJournalDefault>

(mặc định là true)

true

write operations chỉ ack sau khi "majority of the voting members" đã "journal" vào disk

bắt buộc các member đều phải enable journaling

false

write operations ack khi data đc ghi vào memory (không đảm bảo đã ghi xuống disk)

không bắt buộc member phải enable journaling

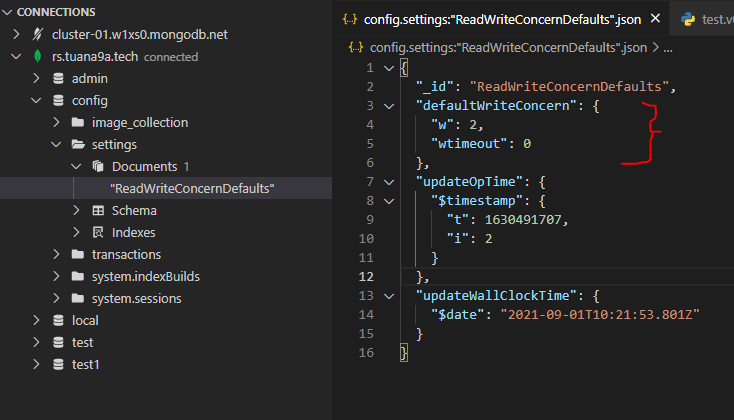
You cannot run transactions on a sharded cluster that has a shard with writeConcernMajorityJournalDefault set to false

(such as a shard with a voting member that uses the in-memory storage engine).

# Reference

## Default Write Concern

set default, global write concern

<https://docs.mongodb.com/v4.4/reference/command/setDefaultRWConcern/#mongodb-dbcommand-dbcmd.setDefaultRWConcern>

CAUTIONS: command có từ phiên bản 4.4

// set default cho cả đọc và ghi

db.adminCommand({

setDefaultRWConcern: 1,

defaultReadConcern: { level : "majority" },

defaultWriteConcern: { w: "majority" }

})

// chỉ set default cho ghi

db.adminCommand({

setDefaultRWConcern: 1,

defaultWriteConcern: { w: 2 }

})

// kiểm tra default read write concern

<https://docs.mongodb.com/v4.4/reference/command/getDefaultRWConcern/>

db.adminCommand({

"getDefaultRWConcern": 1

})

## Local Database

<https://docs.mongodb.com/v4.4/reference/local-database/>

### startup\_log

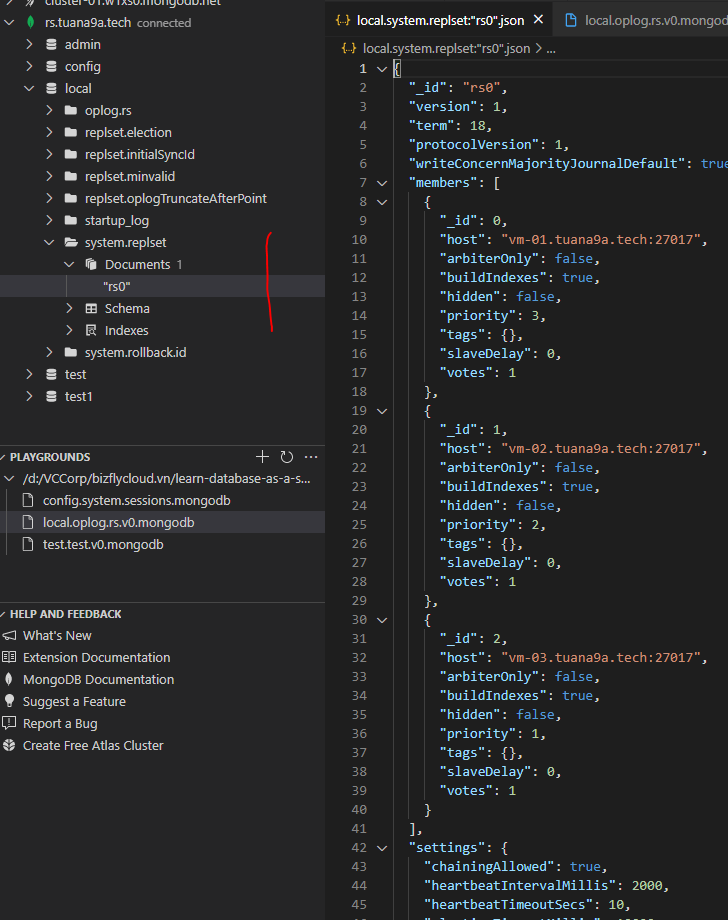
chứa log start up



### system.replset

collections chứa config của replica set

chứa 1 document config là kết quả trả về khi chạy lệnh rs.config() or rs.conf()



### replset.election

chứa nội dung việc "bầu cử" primary

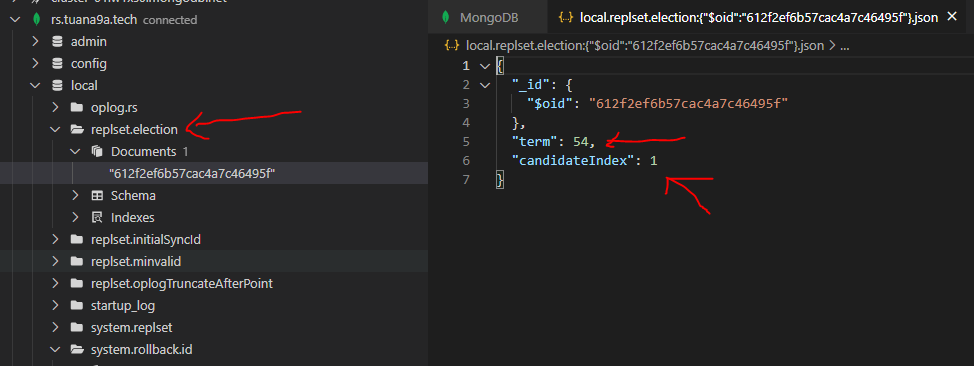
{

term: x // kỳ "bầu cử" thứ x, với mỗi lần "election" term sẽ tăng thêm 1

candidateIndex: x // "ứng viên" thứ x "đắc cử"

}

VD: ảnh dưới



"kỳ bầu cử" gần nhất lần thứ 54 và "ứng viên" đắc cử là ứng viên thứ 1 (tính từ 0)

members[0] vm-01.tuana9a.tech

members[1] vm-02.tuana9a.tech

members[2] vm-03.tuana9a.tech

tức hiện tại vm-02.tuana9a.tech đang là primary

vậy lệnh này sẽ giúp ta biết được hiện tại node nào đang là primary từ bất kì node nào

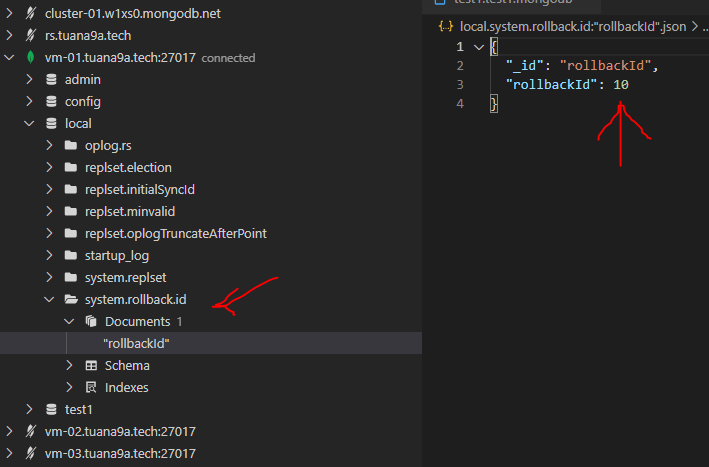
### oplog.rs

là collection chứa oplog đã nêu [ở trên](#_qjdufligy1xp)

### system.rollback.id

chứa document chỉ số lần rollback của ? tại local của một node

hiện tại chưa thấy giá trị này có ý nghĩa lắm



## Command

<https://docs.mongodb.com/v4.4/reference/command/nav-replication/>

change sync source

<https://docs.mongodb.com/v4.4/reference/command/replSetSyncFrom/#mongodb-dbcommand-dbcmd.replSetSyncFrom>

tạm thời khiến node hiện tại chuyển "sync source" sang một node mới

sẽ có trường hợp câu lệnh trả kết quả { ok:1 } nhưng không thay đổi source

do vậy lệnh này chỉ nên dùng cho việc test còn thực tế nên để node tự chọn sync source

<https://docs.mongodb.com/manual/core/replica-set-sync/#replication-sync-source-selection>

việc chuyển sẽ được "lâu" nếu sync source đáp ứng các yêu cầu của việc lựa chọn sync source

db.adminCommand( { replSetSyncFrom: "hostname<:port>" })

VD: db.adminCommand({ replSetSyncFrom: "vm-02.tuana9a.tech" })

## Data Sync

connection

các node trong một replica set sẽ có nhiều connection với nhau (đôi một)

khi một node die

các node còn lại gửi heartbeat không thấy phản hồi các node này sẽ đóng connection với node bị die

do vậy tất cả TCP connection tới node này sẽ được đóng dần lại và sau đó là đóng hẳn tức việc sync bị tắt hẳn

khi một node die được bật trở lại, nó sẽ gửi heartbeat tới tất cả các node còn lại

và sau đó các connection trở lại bình thường với node này

consistency

với mọi trường hợp khi dữ liệu không nhất quán các node sẽ cố tìm ra điểm chung nhiều nhất và mới nhất (major commit point)

cụ thể là phụ thuộc vào oplog "bền" nhất ( last Durable OpLog time ) để từ đó đưa ra quyết định cuối cùng (rollback, update, … )

<https://github.com/mongodb/mongo/blob/master/src/mongo/db/repl/README.md#commit-point-propagation>

## Links

<https://github.com/mongodb/mongo/blob/master/src/mongo/db/repl/README.md>

<https://github.com/mongodb/mongo/blob/master/src/mongo/db/repl/README.md#heartbeats>

các mô hình triển khai cụm mongo

<https://docs.mongodb.com/manual/core/replica-set-architecture-three-members/>

<https://docs.mongodb.com/manual/core/replica-set-architecture-geographically-distributed/>