# Khái niệm mongoDB

Các thành phần chính của mongoDB

* Database

Database là một ô chứa dữ liệu ở mức vật lý, mỗi database bao gồm nhiều collection

* Collection

Collection là một nhóm các văn bản (tương ứng với các bảng trong cơ sở dữ liệu quan hệ), mỗi collection chỉ thuộc về một database duy nhất. Nhưng điểm khác biệt đó là các collection không có sự rằng buộc về cấu trúc của document, nó chỉ là 1 cách gom lại các document tựa tựa nhau tùy theo người quản trị hệ thống. Và các collection không có sự rằng buộc với nhau.

* Document



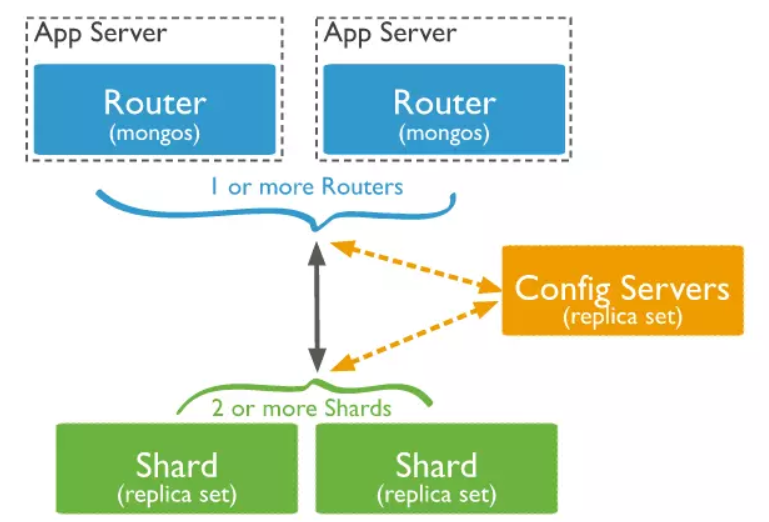
## Cách mongo nhân bản và tổ trức lưu trữ nhân bản

Kỹ thuật nhân bản của mongo đơn giản chỉ là sao một đối tượng dữ liệu ra nhiều máy chủ khác nhau. Trong nhóm máy chủ này có 1 máy chủ chính gọi là master còn các máy chủ khác là máy chủ thứ cấp (gọi là slaver). Máy chủ master đóng vai trò thực hiện các tác vụ đọc và ghi, trong khi các máy chủ slaver chỉ có vai trò backup dữ liệu và tác vụ đọc. Mongo có một cơ chế chuyển đổi dự phòng nếu máy master crash, một trong các máy slaver sẽ lên làm master thông qua thực hiện hàm bầu cử. (Chi tiết mình sẽ nói sau nếu có thời gian) Một cụm máy như này trong kiến trúc của mongo sẽ có 3 máy, gọi là 1 replicaset. Một replica set thông thường sẽ có 1 primary(master) và 2 secondary(slave). Hoặc 1 replica set sẽ có 1 primary, 1 secondary và 1 arbiter. Vai trò của arbiter và lí do vì sao xuất hiện thằng này, mình sẽ nói ở một bài viết khác nhé

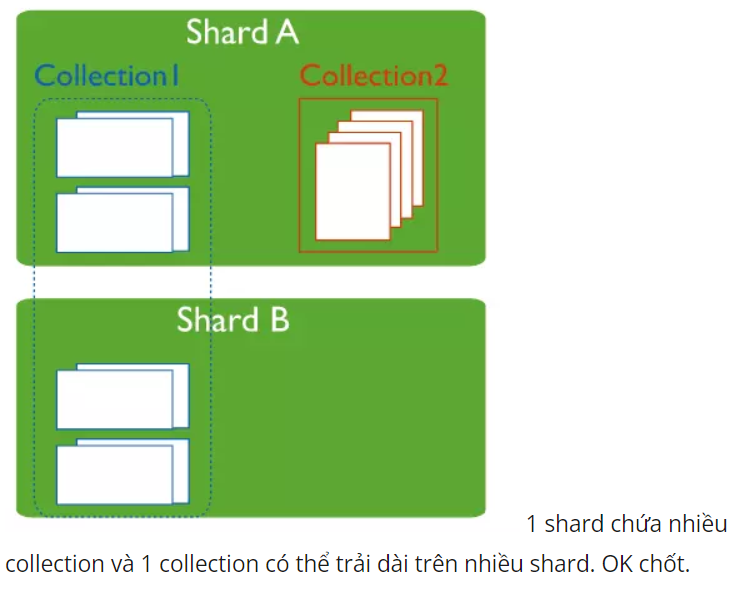
## Cách mongo cắt nhỏ dữ liệu để lưu trữ

Để chia nhỏ một khối dữ liệu lớn ra nhiều các máy trạm khác nhau, mongo có một cơ chế là sharding. Sharding là một kỹ thuật cho phép các bản ghi trong một cơ sở dữ liệu phân tán ra nhiều máy chủ khác nhau. Điều này giúp cho khả năng mở rộng, lưu trữ và xử lỹ của hệ thống. Một sharded cluster bao gồm các thành phần sau:

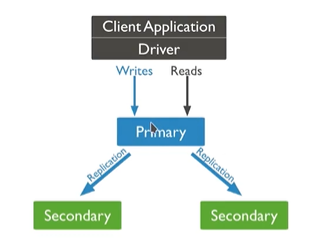
* shard: mọi người hãy hiểu shard là đơn vị quản lý dữ liệu của sharded cluster, mỗi shard thường là 1 replica set
* mongos: đóng vai trò như một bộ định tuyến, là interface đối với người dùng, người dùng chỉ giao tiếp với mongos và không cần quan tâm bên trong có cái gì
* config server: là một máy chủ mongo (mongod) chứa các thông tin cấu hình của cluster, như document nào nằm ở shard nào chẳng hạn. Từ bản 3.0 trở đi, config server cũng là 1 replicaset Dưới đây là hình vẽ mình lấy trên trang chủ của mongo cho mọi người dễ hình dung:



Đấy, đại khái cấu trúc tổng thể nó là như thế. Vậy chính xác các shard chứa gì, các shard chứa các collection hoặc chứa 1 phần của collection, rồi tất cả đống n chứa đấy được nhân lên làm 3 để thành 1 replica set.



# Tại sao cần mongoDB



Vì node primary có nhiệm vụ đọc và ghi cho APP => 2 node còn lại có nhiệm vụ backup, lưu log cho con primary

* 3 con node này phải là 3 con node riêng, đặt cách xa nhau để tránh thảm họa

Node phụ cũng có thể sao chép từ node phụ khác

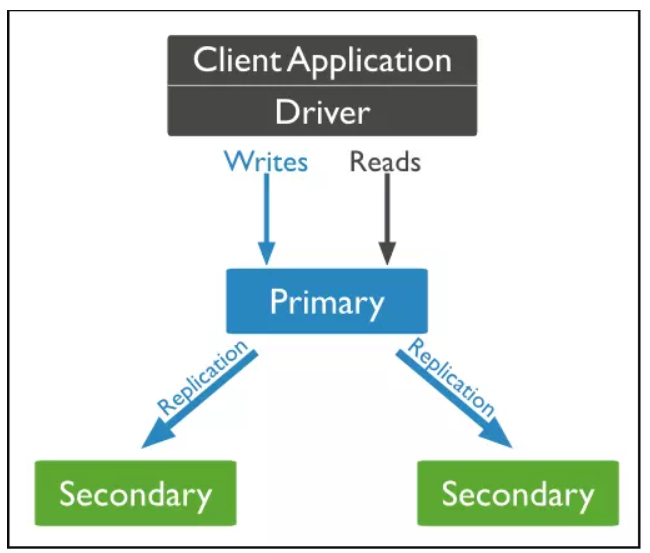
# Set up replica set trong mongodb

Replica set trong MongoDB là một group các mongod processes để duy trì cùng một cơ sở dữ liệu. Replica set cung cấp khả năng dự phòng và tính sẵn sàng cao.

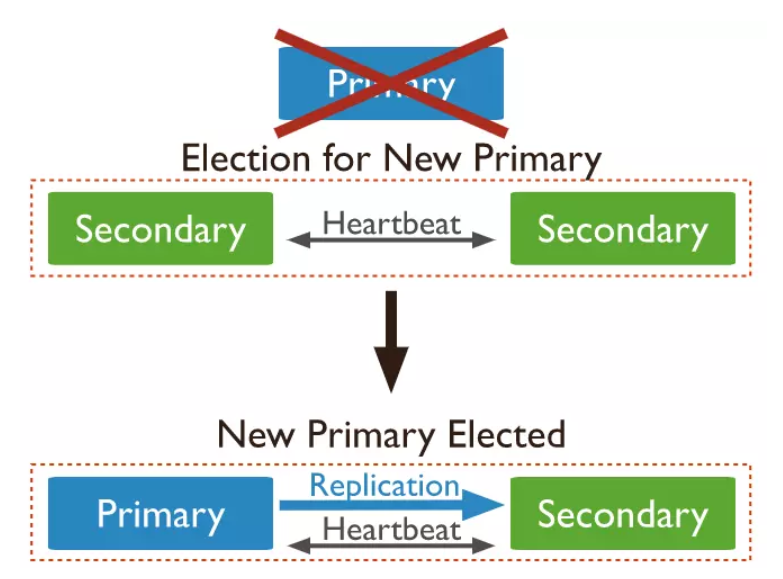
**Tính dự phòng và sẵn sàng cao**

* Replication cung cấp tính dự phòng và nâng cao tính khả dụng của mongoDB. Với nhiều bản sao chép trên nhiều database server khác nhau, replication nâng cao mức độ chịu lỗi chống lại việc mất dữ liệu trên signle database.
* Trong một số trường hợp, replication có thể cải thiện khả năng đọc dữ liệu vì client có thể đọc dữ liệu trên nhiều database server khác nhau. Duy trì các bản sao dữ liệu làm tăng tính cục bộ và khả dụng của dữ liệu.
* Các replica cũng có thể duy trì cho các mục đích như khôi phục dữ liệu, sao lưu.

**Replication trong MongoDB**

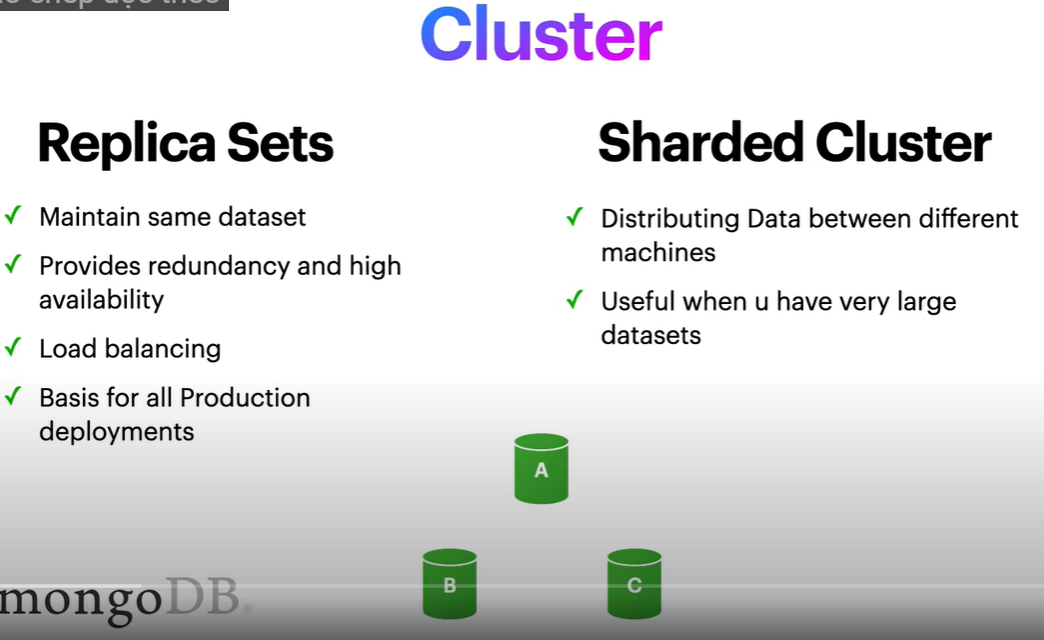


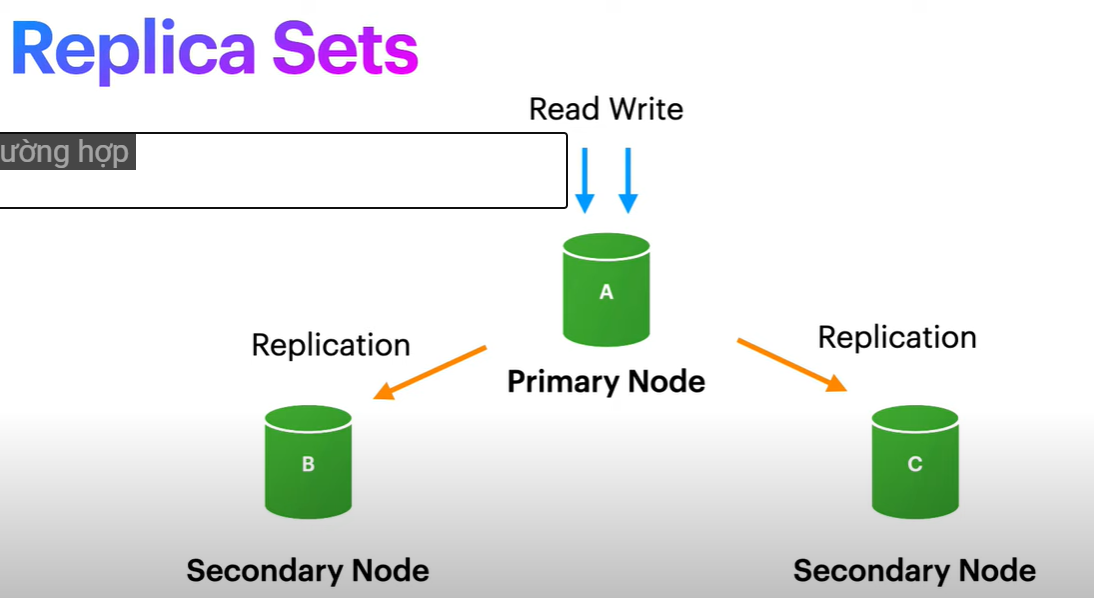
* Replica set là một group các mongod instances để duy trì cùng một bộ cơ sở dữ liệu. Trong replica set bao gồm một số node mang dữ liệu, trong các nút này chỉ duy nhất một nút là nút chính để ghi dữ liệu trong khi các nút còn lại là các nút phụ được sao chép dữ liệu từ nút chính. Nếu nút chính bị lỗi một trong những nút phụ sẽ được thay thế thành một nút chính mới.



Mongodb là 1 cụm cluster, có 2 loại cluster cho mongodb

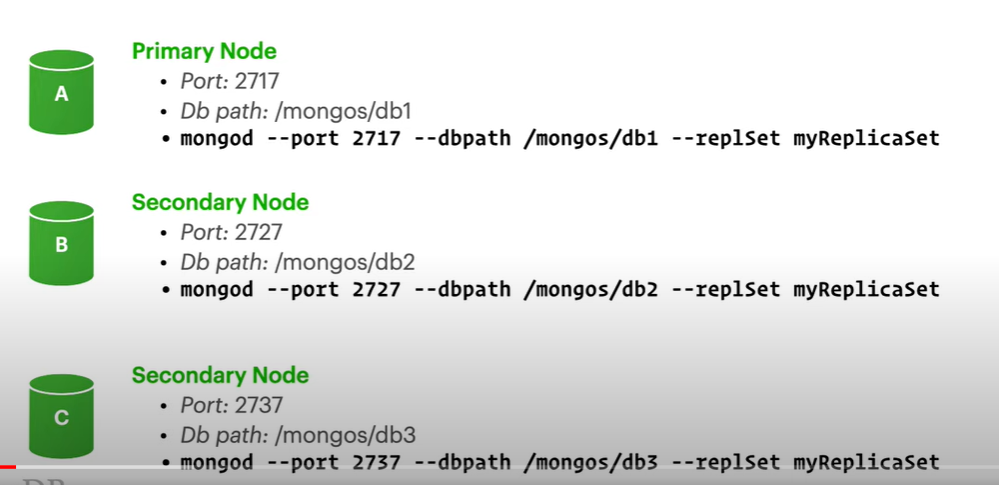
* Replica Sets
* Sharded Cluster





* Con primary sẽ có nhiệm vụ đọc ghi
* 2 con còn lại: backup dữ liệu và task vụ đọc

**Tạo replicas**



mongod: run command

--port 2717: custom port

--dbpath /mongos/db1 : đường dẫn storage

--replicas : sau nó là tên replicas

## Command mongod

* Trạng thái replica Set

rs.status()

* Khởi tạo 1 replica set trên primary node

rs.initiate

* Add 1node tới replica

rs.add(“localhost:2727”)

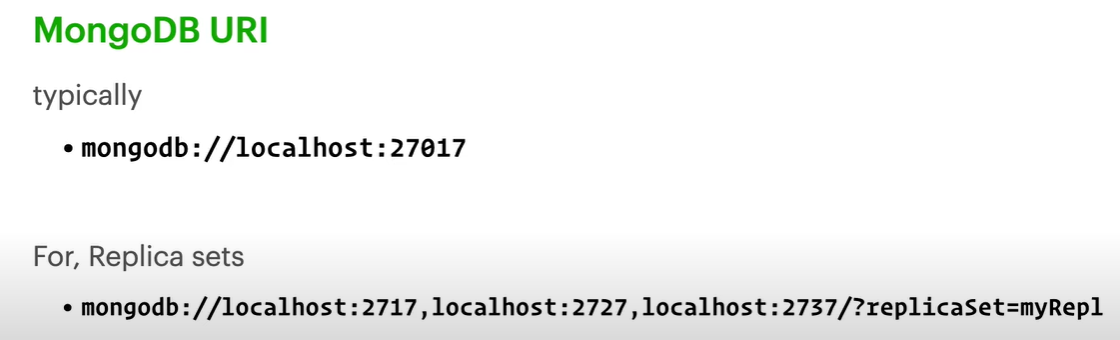
* Để add 1 machine mongodb khác tới replicas

Rs.add([$IP]:[$port]) port: port hđ của mongodb trên intance đó

* Để remove 1 intance khỏi replicas

Rs.remove

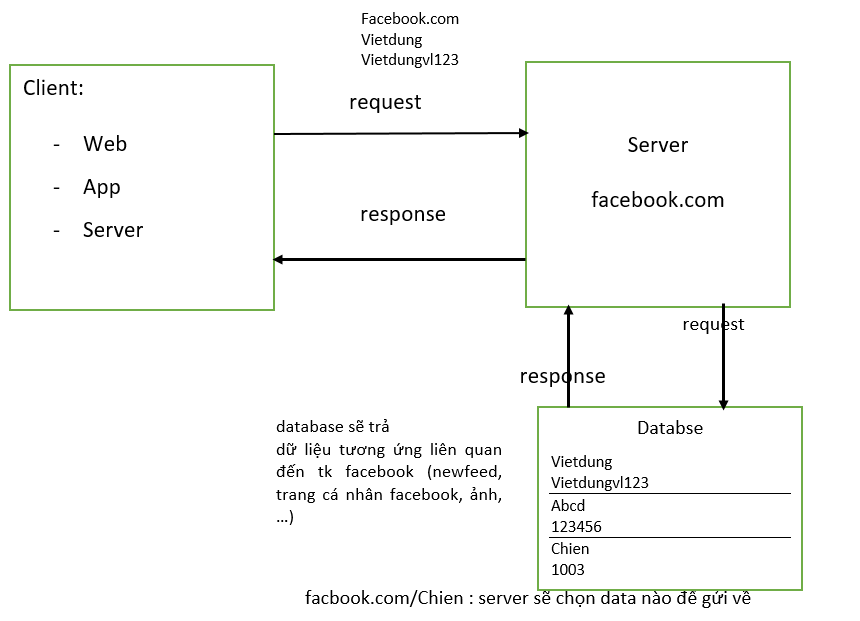
* MongoDB URI



Cái đầu là node1:port (primary), cái 2 3 là của second

Cuối cùng là tên replicas

# Phần 2: mongoDB là gì



ví dụ: con server là facebook.com

người dùng request đến server: facebook.com gửi kèm user account/pass

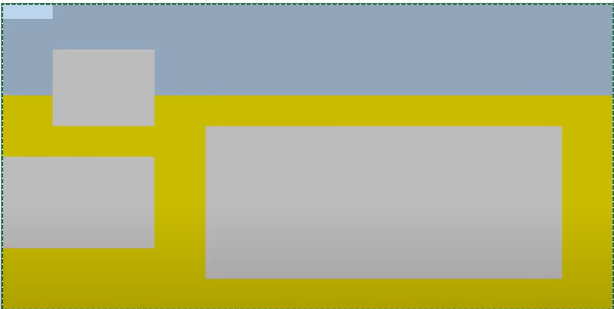
con server sẽ nhận vào 2 cái data này username / password => vậy server sẽ làm gì vs 2 data này. Nó sẽ đối chiếu 2 data này với database, nếu khớp database sẽ trả dữ liệu về cho server -> tới client (newfeed của facebook)

database là kho chứa dữ liệu,

=> server cần có database, chứ có cái server ko thì làm dk j

Server sẽ trả về dữ liệu mà nó có khung bảng thế này

Mỗi 1 account sẽ có khung giao diện giống nhau, nhưng nội dung bên trong là khác nhau



New Feed

Nhưng có phải 1tr ng dùng, server sẽ lưu trữ cả 1tr khung newfeed khác nhau ko

* Câu trả lời là Không

Sẽ chỉ có đúng 1 khung Newfeed (giống như chỉ cần 1 cái index.html) – web Tĩnh

Và các dữ liệu trong khung là khác nhau thôi, và dữ liệu trong khung là từ database đổ về

* Database cực kỳ quan trọng, dữ liệu ng dùng ở hết đây

Có 2 trường phái database

* SQL : MySQL, SQLserver, POSTGRESQL
* NoSQL : MongoDB, Redis, Oracle

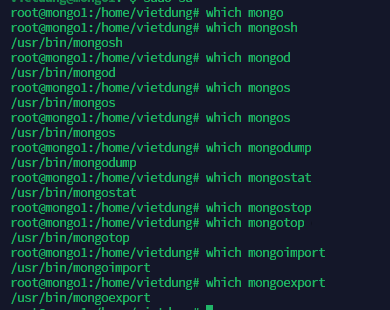


## Làm sao để biến 1 con server thành database

Cài mongoDB => thành DB

DB chạy rồi, vậy làm sao để thêm dữ liệu vào database

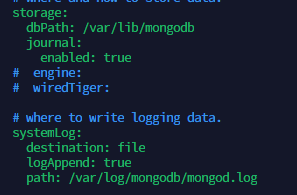
* Câu lệnh: mongo shell , tạo, truy xuất, thêm sửa xóa (nhưng ko trực quan, khó nhìn)
* Tool: Robo Mongo, mongo compass : hỗ trợ giao diện UI để tương tác vs database (dễ nhìn)



mongo, mongosh : command shell, nhưng phiên bản mới n bỏ **mongo shell** r

mongod : cũng là command shell, nhưng sẽ command bên ngoài

đây là 1 số thông tin file của thằng mongoDB, ta có thể tùy chỉnh, thay đổi,

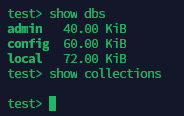


tuy nhiên vẫn lưu ý 1 điều rằng



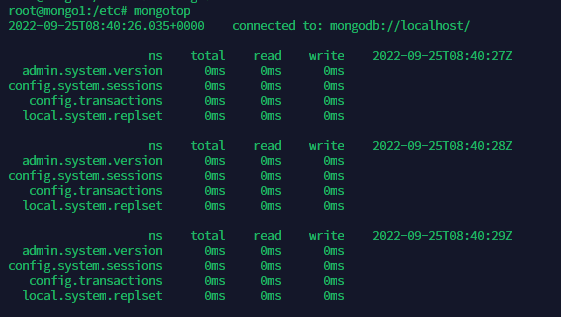
Nếu có thay đổi đường dẫn của mongodb, ta phải trace lại quyền user/group cho thằng mongodb

# CMD

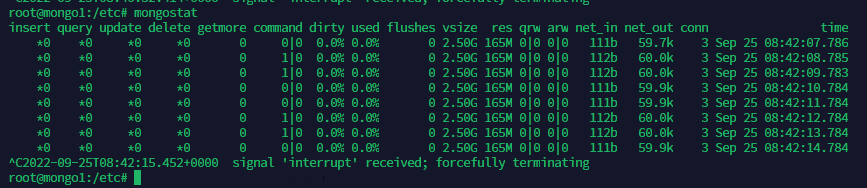


Chưa có colections nào

Mongotop : thông tin top mongoDB

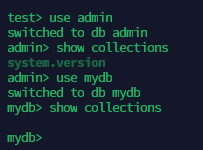


mongostat



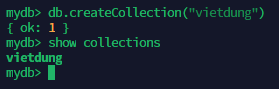
Khi tạo 1 DB mới thì chưa có collection gì

Collection tương tự như tables



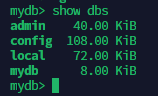
* Tạo 1 collection

db.createCollection("vietdung")



Vì db mydb vừa được tạo 1 collection, nên nó đã có dữ liệu

* Show dbs nên nó đã được hiện thị

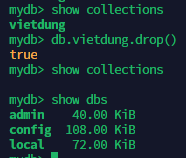


Những DB ko có dữ liệu nó sẽ không được hiển thị

* Xóa 1 collection

db.vietdung.drop()

vietdung : tên của collection trong 1 DB

 Khi mà DB trống thì n ko được hiển thị

* Import dự liệu vào DB

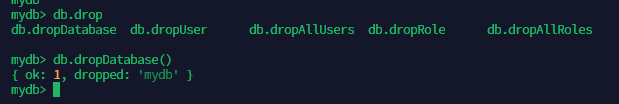
db.vietdung.insert( { "title": "deptrai", "realease": 1999 } )

db.vietdung.find() : show thông tin

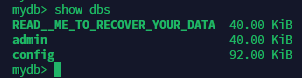


* Xóa Database

Đang ở trong database mydb, xóa DB này



Thực sự là DB đã được xóa



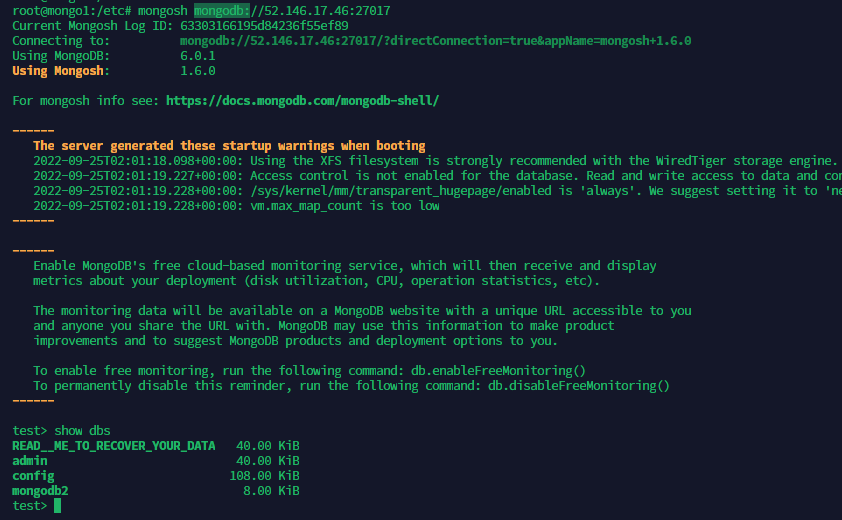
Từ con mongodb1 access vào mongodb2

Mongodb2 IP :52.146.17.46

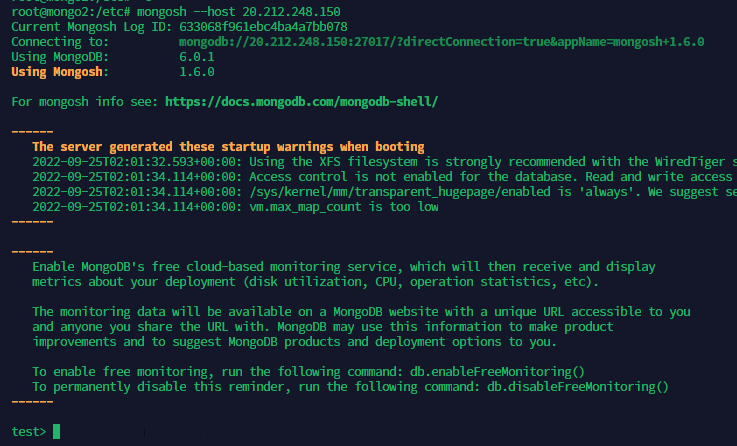
root@mongo1: # mongosh mongodb://52.146.17.46:27017

mongosh --host 20.212.248.150

mongosh --host mongodb2 (nếu chỉnh ở /etc/hosts)



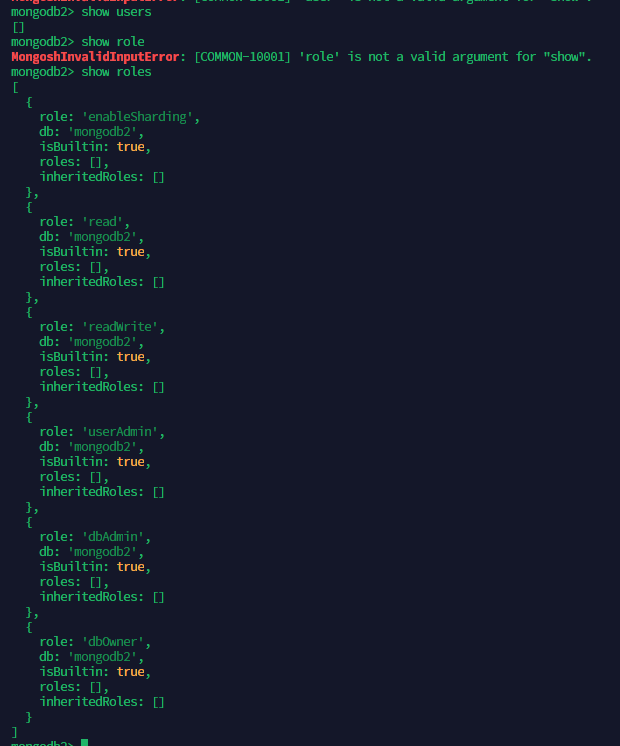
Oke



# User, role, authen

show users

show roles



* Tạo user

use admin

db.createUser(

{

user: "vietdung",

pwd: "vietdungvl123",

roles: [ { role: "userAdminAnyDatabase", db: "admin" } ]

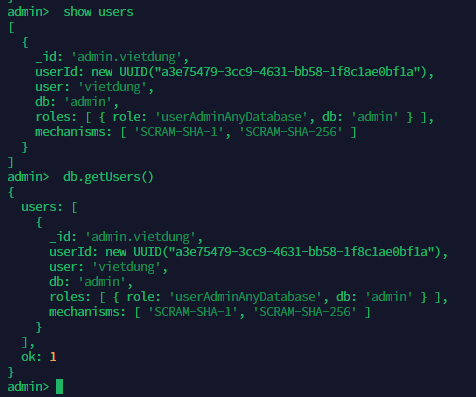
}

)

* Show user

show users

db.getUsers()

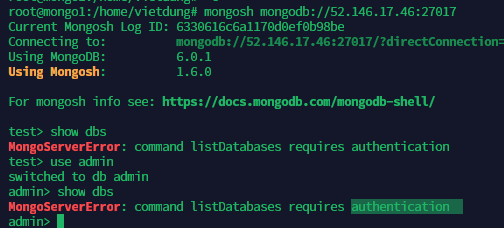


* Bật authentication

security:

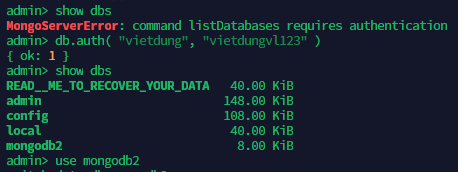
authorization: "enabled"

bây giờ từ con server khác access vào



Vừa là tạo user và bật authen trên con mongo2, giờ từ con 1 access vào

db.auth( "vietdung", "vietdungvl123" )



* Tạo 1 user DB khác

use demo

db.createUser(

{

user: "justAUser",

pwd: passwordPrompt(), // or cleartext password

roles: [ { role: "readWrite", db: "demo" },

{ role: "read", db: "finances" } ]

}

)

* Đổi password user

db.changeUserPassword( "vietdung", "admin")

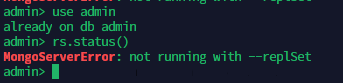
user vietdung sẽ đổi password thành admin

# Replica Set mongodb

Vào DB admin kiểm tra xem có replicas nào chưa

* Kiểm tra replica Set DB

rs.status()



Oke chưa

# Thiết lập Replicas

**B1:** install mongodb

**B2:** edit config IP binding 0.0.0.0

**B3:** thiết lập host trên 3 con

nano /etc/hosts

52.146.17.42 mongodb1

52.146.17.46 mongodb2

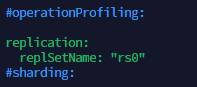
20.212.248.150 mongodb3

**B4:** thiết lập replicaSet trước, authen sau

nano /etc/mongod.conf

replication:

replSetName: "rs0"



Vào mongosh,

Use admin

rs.initiate( {

\_id : "rs0",

members: [

{ \_id: 0, host: "mongodb1:27017" },

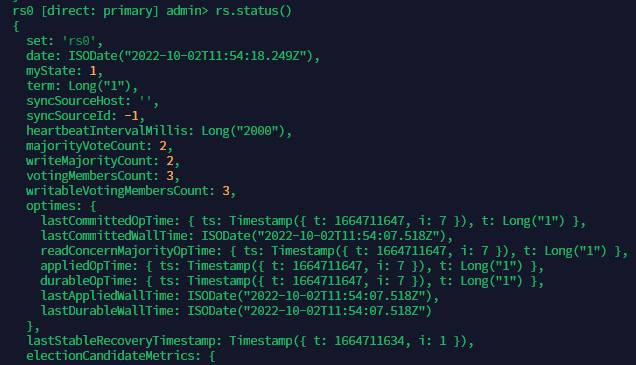
{ \_id: 1, host: "mongodb2:27017" },

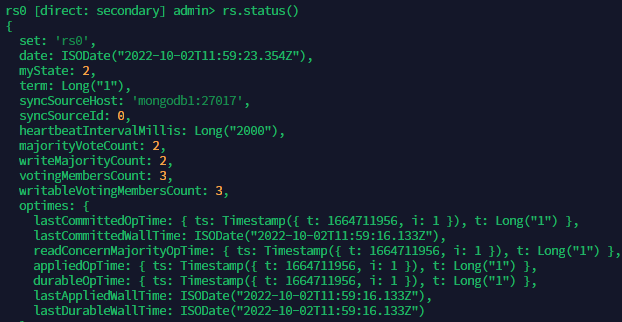
{ \_id: 2, host: "mongodb3:27017" }

]

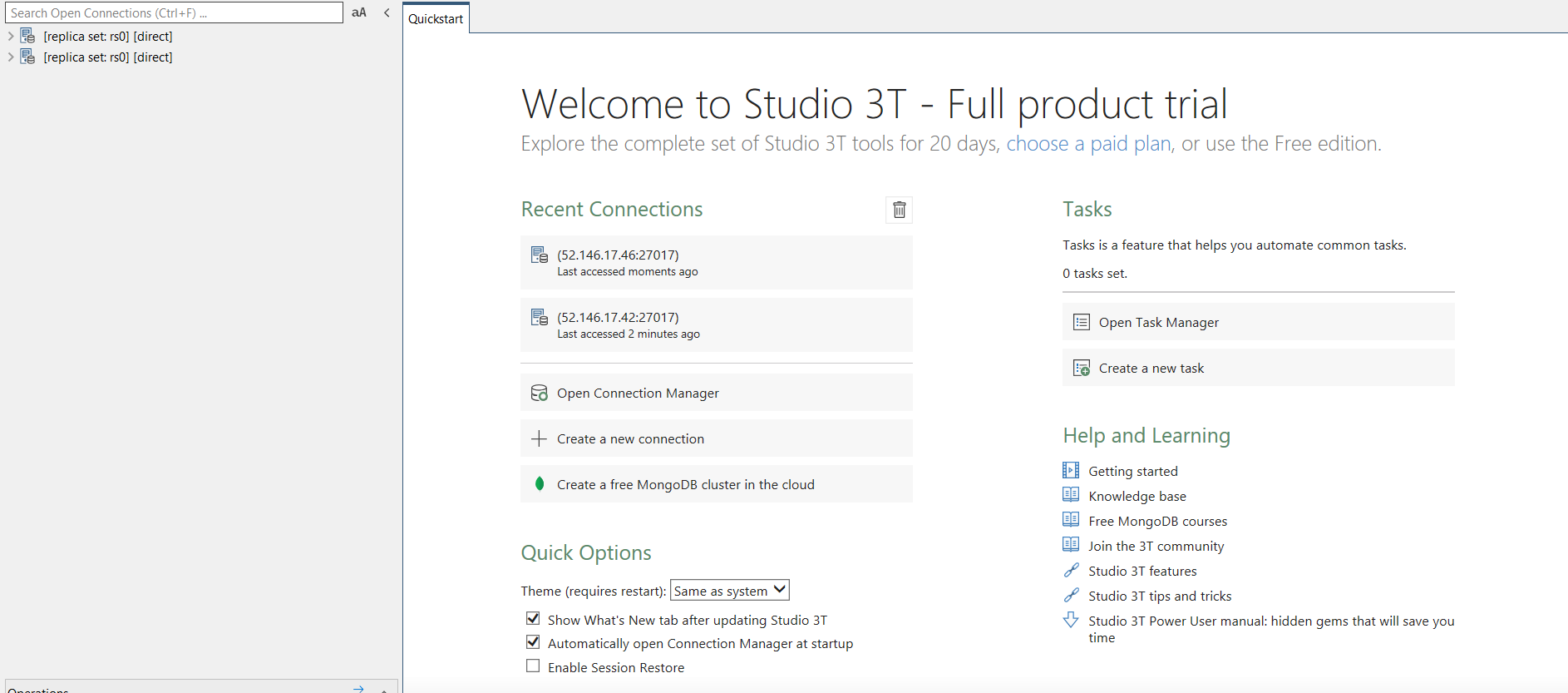
})

Mongodb1 & 2 & 3 đã được định nghĩa trong file host

 oke trên con 1 hiện primary

 con 2 3 là second

Bgio tạo 1 DB trên con primary thì 2 con second cũng sẽ thấy



Connect Robo 3T bgio n cũng chỉ hiện replicas DB

**B5:**

tạo user role { “root” } trong db admin : ko cần tạo user trên 2 con second, tý n tự replicas

Và tạo thêm 1 DB riêng để dùng chứ ko dùng DB admin, DB admin chỉ để authen

* User admin DB admin

use admin

db.createUser(

{

user: "admin",

pwd: "admin",

roles: [ { role: "root", db: "admin" } ]

}

)

Tạo 1 DB cho việc dùng

use db\_fnb

db.createUser(

{

user: "vietdung",

pwd: "vietdungvl123",

roles: [ { role: "userAdmin ", db: "db\_fnb" } ]

}

)

Lưu ý: mỗi DB có 1 role riêng

Check bằng lệnh **Show roles**

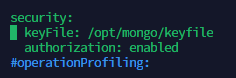
DB admin có nhiều role nhất

Rồi oke, chỉ cần tạo trên primary, nó sẽ tự ánh xạ trên 2 con second

**B6:** tạo file authen cho n bảo mật, trên primary

openssl rand -base64 741 > /opt/mongo/keyfile

* copy file này tới 3 con
* cho quyền 600
* chown mongosh:mongosh

 để ở chỗ nào thì keyFlie trỏ đến đấy

security:

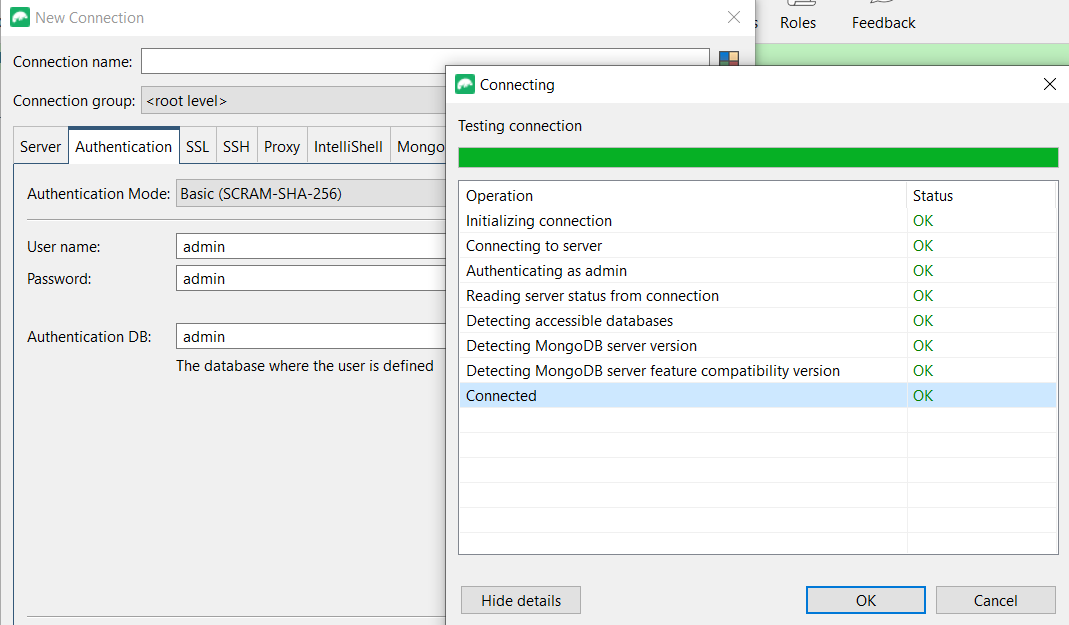
keyFile: /opt/mongo/keyfile

authorization: enabled

copy file: scp keyfile [vietdung@20.212.248.150:/home/vietdung](mailto:vietdung@20.212.248.150:/home/vietdung)

ko copy thẳng được vì file kia để quyền mongodb rồi

Vào robo 3T test connection



Có thể authen từ 1 server khác

mongosh db -u vietdung --host 52.146.17.42 –p

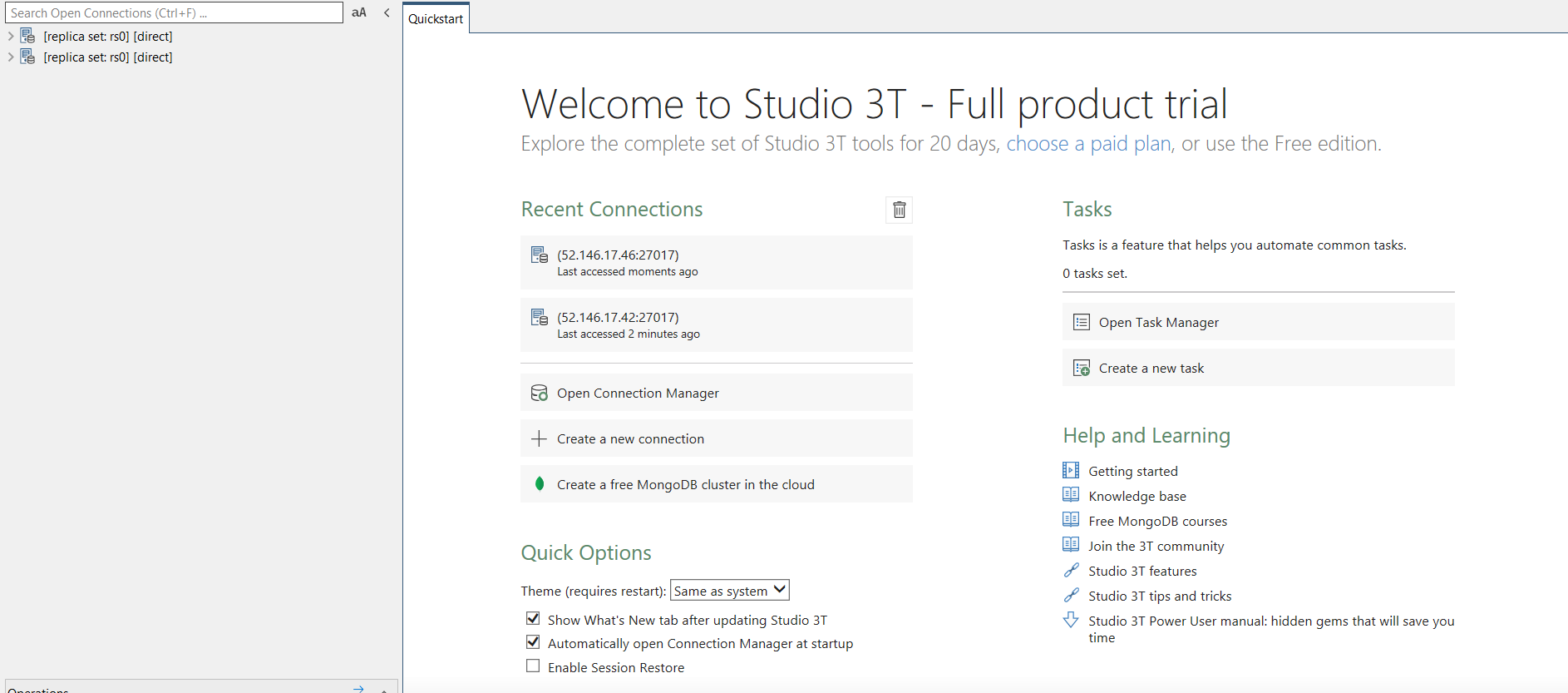
mongosh mongodb://mongodb1,mongodb2,mongodb3:27017

sẽ tự động access vào con primary

* db: tên của DB
* u vietdung : tên user
* --host : tên server mongodb
* -p : nhập password



Oke luôn



Access DB n sẽ hiện ra tên replicas thôi

Bây giờ restart DB con primary, thì con primary sẽ nhảy sang con DB khác

# Redis

**B1:** install redis

**B2:** install redis-sentinel

apt-get install redis-sentinel

chown redis:redis /etc/redis/sentinel.conf

chown redis:redis /var/log/redis/redis-sentinel.log

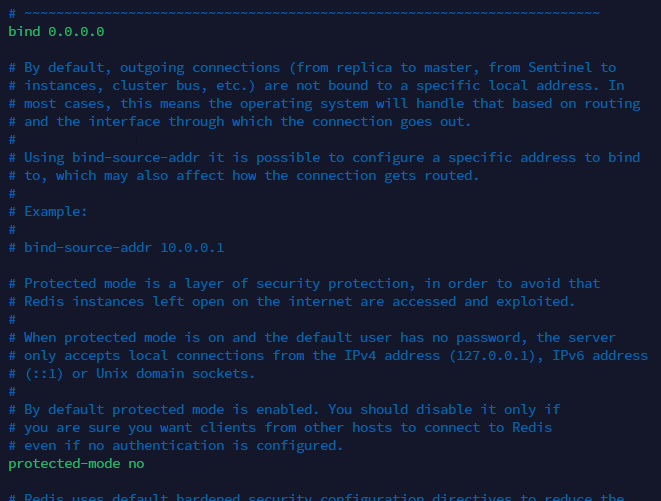
* Stop tạm redis sentinel

systemctl stop redis-sentinel

**B3:** check ufw, cho phép port 22, 6379, 26379

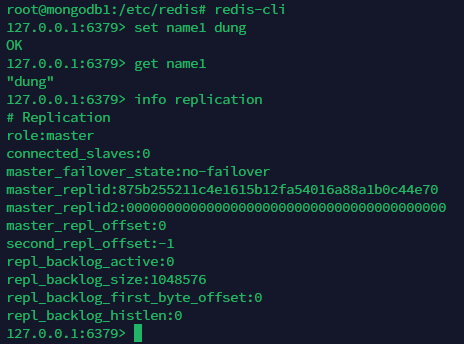
**B4:** /etc/redis/redis.conf

* Bind IP
* Port
* protected-mode no



systemctl restast redis-server

check: set này tương tự set cache, dạng key - vault



Get thông tin replicas

Role: master and slave: 0

**II.** Insall tiếp trên con redis 2 và 3

Biến con 1 thành con master, con 2 3 sẽ là slave

Vào con 2 3, vẫn chạy các bước như trên, tuy nhiên ta sẽ thêm 1 bước

nano /etc/redis/redis.conf

slaveof 52.146.17.42 6379 (thêm vào cuối file)

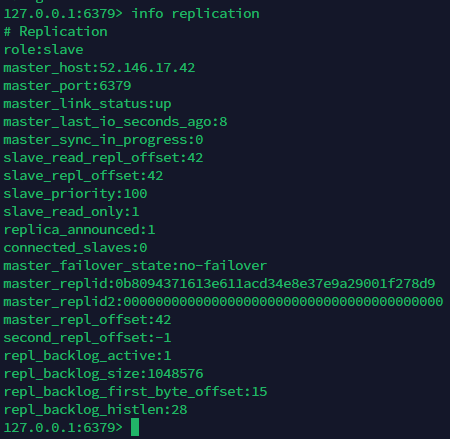
52.146.17.42 : là IP của con redis 1 (master)

Restart lại redis

Check:



Cái name1 này vừa được set ở con master 1



Đã có thông tin replicaset với IP master là IP của con redis 1 (master)

 trên con master

Lưu ý: tgian lưu cache của con redis có giới hạn

**B6: config sentinel redis server**

Master node (redis master)

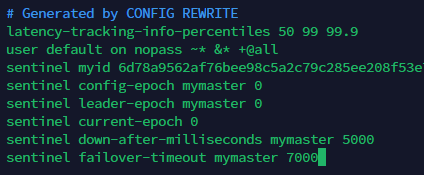
1. config /etc/redis/sentinel.conf

bind IP: bind 0.0.0.0 (hoặc để IP chính nó và 127.0.0.1)

protected-mode no

port: 26379 – port cho sentinel

sentinel monitor mymaster 52.146.17.42 6379 2



* down time
* failover time

sentinel config-epoch mymaster 0

sentinel down-after-milliseconds mymaster 5000

sentinel failover-timeout mymaster 7000

config sentinel service

nano /usr/lib/systemd/system/redis-sentinel.service

hoặc nano /etc/systemd/system/sentinel.service (check xem file này ở đây)

ExecStart=/usr/bin/redis-server /etc/redis/sentinel.conf --sentinel

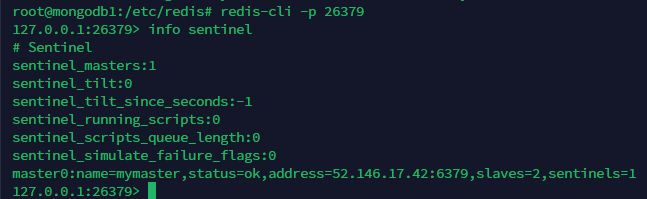
systemctl daemon-reload

systemctl start redis-sentinel

systemctl restart redis-sentinel

systemctl status redis-sentinel

redis-cli -p 26379



Info: sentinel

Master: 52.146.17.42

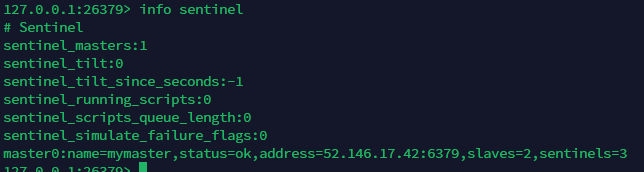
Slave: 2 – sentinel: 1

Bây giờ vào con 2 3 làm tương tự

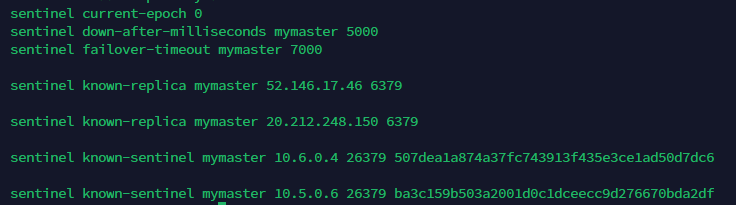
sentinel monitor mymaster 52.146.17.42 6379 2

vẫn IP con master

khi cài xong con 2 và 3 thì vào con master đã hiện 3 sentinel

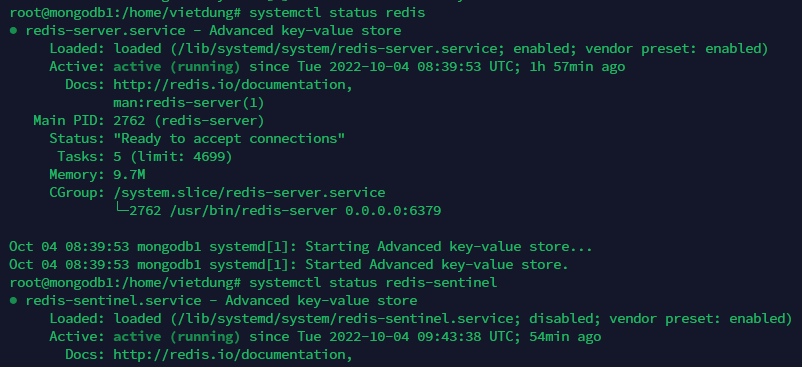


Check file cat /etc/redis/sentinel.conf



Đã tự động gen các IP, info của 2 con slave

# Lưu ý lệnh systemctl status



Bình thường khi ta cài các service, ta có thể check status bằng tên của n,

Vd systemctl status redis, systemctl status redis-sentinel

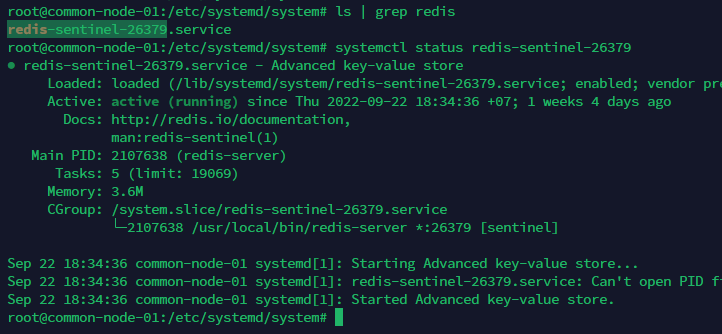
Thực chất là nó nằm trong file  **/usr/lib/systemd/system/redis-sentinel.service**

Cho **redis-sentinel** hoặc **/etc/systemd/system/redis.service**

Có 1 số trường hợp nó custom cái này

Khi mà trong file systemd nó ko phải là **redis-sentinel** mà là **redis-sentinel-26379**

Thì check status phải là **redis-sentinel-26379**



# Install kafka

<https://kafka.apache.org/quickstart#connectconfigs_plugin.path>

install kafka

* start kafka và zookeeper

bin/zookeeper-server-start.sh config/zookeeper.properties

bin/kafka-server-start.sh config/server.properties

* tạo 1 topic

bin/kafka-topics.sh --create --topic quickstart-events --bootstrap-server localhost:9092

* check topic

bin/kafka-topics.sh --describe --topic quickstart-events --bootstrap-server localhost:9092

ghi 1 số event vào topic

$ bin/kafka-console-producer.sh --topic quickstart-events --bootstrap-server localhost:9092

This is my first event

This is my second event

Đọc event

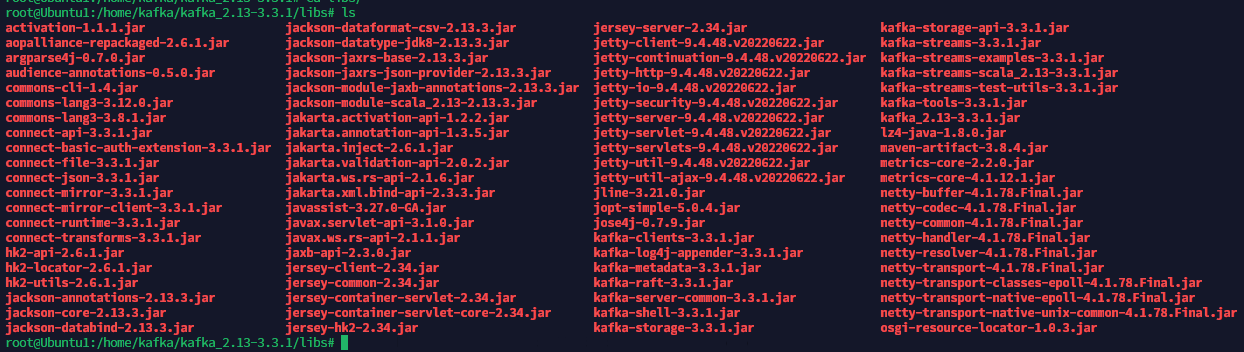
$ bin/kafka-console-consumer.sh --topic quickstart-events --from-beginning --bootstrap-server localhost:9092

## Kafka connect

Tạo 1 plugin đơn giản cho connect-standalone

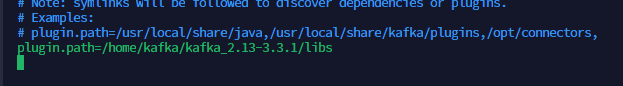
echo "plugin.path=libs/connect-file-3.3.1.jar"

tức là tạo 1 file connect-file-3.3.1.jar trong file kafka/libs

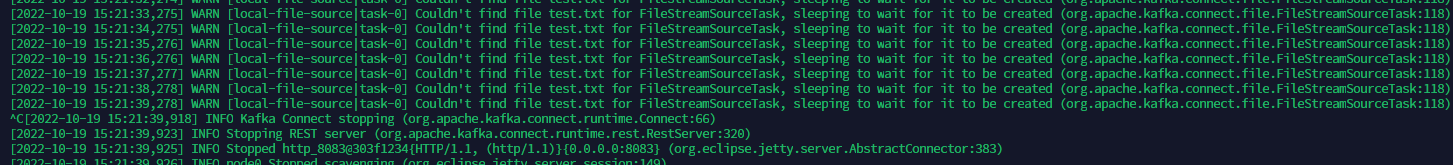


Bgio edit file connect-standalone, thiết lập plugin trỏ vào đường dẫn file .jar

nano config/connect-standalone.properties

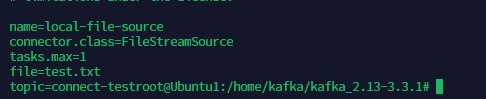


bin/connect-standalone.sh config/connect-standalone.properties config/connect-file-source.properties config/connect-file-sink.properties



Chạy lên trên lỗi ko thấy file test.txt vì file config/connect-file-source.properties nó đang gọi đến file test.txt

2 file source và file sink cho 2 chế độ của kafka



## Start kafka connect port 8083

Nano config/connect-distributed.properties

Thêm

listeners=HTTP://:8083

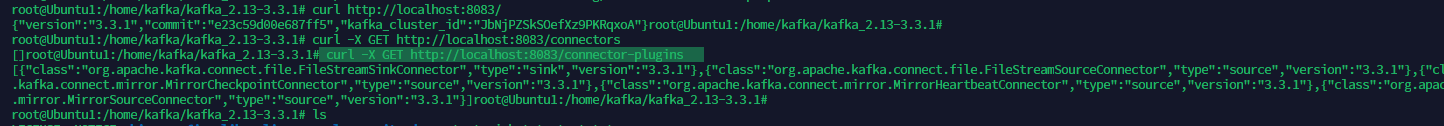
plugin.path=/home/kafka/kafka\_2.13-3.3.1/libs // đường dẫn chứa các plugin

Sau đó khởi chạy Connect Server với lệnh sau:

bin/connect-distributed.sh -daemon config/connect-distributed.properties

* check connect plugin

curl -X GET http://localhost:8083/connector-plugins



bin/connect-distributed.sh config/connect-distributed.properties config/connect-file-source.properties config/connect-file-sink.properties