

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN



TIỂU LUẬN THỰC HÀNH
HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU MONGODB

MÔN HỌC: IS211 - CƠ SỞ DỮ LIỆU PHÂN TÁN
LỚP: IS211.N11
GIÁO VIÊN: ThS. THÁI BẢO TRÂN
NGUYỄN HỒ DUY TRÍ

Nhóm sinh viên thực hiện:

Bùi Nguyên Phương Linh – 20521527
Nguyễn Thành Nhân – 20521701
Ngô Thị Phúc – 20521765
Nguyễn Thị Phương Thảo – 20521936

TP. HỒ CHÍ MINH – 12/2022

MỤC LỤC

PHỤ LỤC

LỜI MỞ ĐẦU

I. GIỚI THIỆU	1
1. MongoDB là gì:.....	2
2. Lịch sử hình thành MongoDB.....	3
3. Tác giả, tổ chức quản lý	4
II. TỔNG QUAN VỀ MONGODB.....	4
1. Các kiểu dữ liệu trong MongoDB.....	6
2. Một số câu lệnh dùng trong MongoDB	8
3. MongoDB hoạt động như thế nào?	9
4. Ngôn ngữ thao tác với dữ liệu.....	10
5. Mô hình lưu trữ	10
5.1. Văn bản (<i>documents</i>).....	10
5.2. Bộ sưu tập (<i>collections</i>)	11
5.3. Thiết kế lược đồ.....	11
5.4. Chỉ mục (<i>Index</i>).....	12
6. Nhân bản (Replication)	13
6.1. Master-Slave Replication.....	14
6.2. Replica Sets	14
7. Cơ chế phân tán.....	15
7.1. Sharding	15
7.2. Replication	20
III. HƯỚNG DẪN CÀI ĐẶT	23
1. Cài đặt trên một máy	23
1.1. Chi tiết cấu hình máy	23

1.2. Các bước thực hiện	23
2. Cài đặt trên cụm máy phân tán.....	28
2.1. Chi tiết cấu hình máy	28
2.2. Các bước thực hiện:	28
IV. THỰC NGHIỆM MÔ PHỎNG PHÂN TÁN.....	45
1. Mô tả bài toán.....	45
2. Cấu trúc dữ liệu sử dụng	45
3. Các bước thực nghiệm	45
3.1. Thực nghiệm truy vấn phân tán	45
3.2. Mô tả thực nghiệm chức năng Arbiter trong cơ chế phân tán Replica Set Member.	52
KẾT LUẬN	
Ưu điểm MongoDB	
Nhược điểm MongoDB	
Kết quả đạt được	
TÀI LIỆU THAM KHẢO	

PHỤ LỤC

Phụ lục bảng

Bảng 1. Các kiểu dữ liệu trong MongoDB	7
Bảng 2. Một số câu lệnh dùng trong MongoDB	8

Phụ lục hình ảnh

Hình 1. Biểu tượng của hệ quản trị cơ sở dữ liệu MongoDB	3
Hình 2. So sánh khái niệm giữa MongoDB và SQL.....	6
Hình 3. Quy trình hoạt động của MongoDB.....	9
Hình 4. MongoDB Drivers.....	10
Hình 5. Minh họa bộ sưu tập.....	11
Hình 6. Mô hình Master – Slave hai nút	14
Hình 7. Mô hình Master – Slave bốn nút	14
Hình 8. Mô hình Replica Sets hai nút	15
Hình 9. Replica Sets – Bầu chọn master mới.....	15
Hình 10. Server chính trở thành server cấp 2.....	15
Hình 11. Cluster “trong suốt” với người dùng.....	16
Hình 12. Sharded Cluster	17
Hình 13. Chunk	19
Hình 14. Chunk Splits	19
Hình 15. Chunk Migration	19
Hình 16. Cơ chế hoạt động Replica Set Member.....	21
Hình 17. Cơ chế Replica Set Member khi có sự cố	21
Hình 18. Cơ chế Replica Set Member có Arbiter	22
Hình 19. Trang chủ tải MongoDB	23
Hình 20. Cài đặt MongoDB	24
Hình 21. Cài đặt MongoDB (Đồng ý các điều khoản và điều kiện)	24
Hình 22. Cài đặt MongoDB (Tùy chọn cài đặt).....	25
Hình 23. Cài đặt MongoDB (Thiết lập cấu hình)	26

Hình 24. Cài đặt MongoDB (Cài đặt môi trường)	26
Hình 25. Hoàn tất cài đặt.....	27
Hình 26. Các bước cài đặt MongoDB Shell	28
Hình 27. Phần mềm máy ảo VirtualBox	29
Hình 28. Tạo mới hộp ảo	29
Hình 29. Thiết lập file nguồn – Type: Linux – Version: Ubuntu (64bit)	30
Hình 30. Ram lí tưởng tối thiểu là 1 GB	30
Hình 31. Tùy chọn cài đặt với hệ điều hành Ubuntu đã tải sẵn	31
Hình 32. Network – Adapter 1	31
Hình 33. Network – Adapter 2.....	31
Hình 34. Network – Adapter 3.....	31
Hình 35. Shared Folders.....	31
Hình 36. Khởi động máy ảo và đăng nhập.....	32
Hình 37. Khởi động Terminal để thao tác lệnh.....	32
Hình 38. Kiểm tra Update – Cài đặt SSH – Fix lỗi cài đặt	33
Hình 39. Fix lỗi not found.....	33
Hình 40. Kiểm tra ID – kết nối PuTTY	34
Hình 41. Đăng nhập kết nối máy ảo Ubuntu thông qua PuTTY.....	34
Hình 42. Nhập khóa và thiết lập để cài đặt MongoDB cho máy ảo	35
Hình 43. Cài đặt gói MongoDB	36
Hình 44. Kiểm tra sau khi cài đặt.....	36
Hình 45. Khởi động MongoDB Shell	37
Hình 46. Cấu hình tập tin mongod.conf.....	38
Hình 47. Chọn Clone máy ảo đã thiết lập	38
Hình 48. Thay đổi tập tin lưu trữ	38
Hình 49. Chọn Full clone	39
Hình 50. Bắt đầu tiến trình clone	39
Hình 51. Thay đổi hostname	39
Hình 52. Thêm chỉ định IP trong tập tin Hosts	40

Hình 53. Ping IP để kiểm tra kết nối.....	40
Hình 54. Kết nối phân tán giữa các máy.....	42
Hình 55. Sao chép tệp mongod.conf thành mongod_arb.conf.....	42
Hình 56. Cập nhật trong tệp mongod_arb.conf.....	42
Hình 57. Tạo và thiết lập “/var/lib/mongodb_Arb”	43
Hình 58. Thiết lập và khởi động Arbiter.....	43
Hình 59. Add Arbiter vào liên kết phân tán	43
Hình 60. Cụm máy phân tán hoàn chỉnh theo cơ chế Replica Set Member	44
Hình 61. Kiểm tra database trên máy trụ sở TP-HCM	45
Hình 62. Tạo database và collection	46
Hình 63. Insert Data vào Database.....	47
Hình 64. Kiểm tra collection và data	47
Hình 65. Cho phép đọc và kiểm tra Database ở chi nhánh Hà Nội	48
Hình 66. Truy vấn phân tán ở chi nhánh Hà Nội.....	48
Hình 67. Truy vấn phân tán có điều kiện – diachi = “TP.HCM”	49
Hình 68. Truy vấn phân tán có điều kiện – diachi = “Ha Noi”	49
Hình 69. Thêm dữ liệu khách hàng có diachi = “Ha Noi” trên máy trụ sở	50
Hình 70. Truy vấn dữ liệu khách hàng có diachi = “Ha Noi” trên chi nhánh Hà Nội.....	50
Hình 71. Insert dữ liệu trên máy Secondary	51
Hình 72. Truy xuất phân tán với role Arbiter	52
Hình 73. Arbiter sẽ không bao giờ thay đổi role.....	52
Hình 74. Sự thay đổi role trong cơ chế replica set member.....	53

LỜI MỞ ĐẦU

Hiện nay, tốc độ khoa học phát triển rất nhanh, đặc biệt trong lĩnh vực Công nghệ Thông tin. Các yêu cầu của các hệ thống phần mềm cần phát triển nhanh, chất lượng tốt, chi phí giá thành giảm, v.v. Lựa chọn hệ quản trị cơ sở dữ liệu là một trong những yếu tố dẫn đến thành công của dự án. Tuy nhiên mỗi loại cơ sở dữ liệu lại có ưu nhược điểm khác nhau, tùy vào bài toán để chọn cơ sở dữ liệu phù hợp. Để đáp ứng yêu tố nhanh, và tức thời trong hệ thống phần mềm, người ta sẽ chọn giải pháp sử dụng cơ sở dữ liệu thời gian thực. Nhưng loại cơ sở dữ liệu này có chi phí vận hành lớn, trong khi đó nhiều dự án chỉ có nguồn kinh phí hạn hẹp. Dữ liệu tăng lên rất nhanh, vượt qua giới hạn xử lý của các hệ quản trị cơ sở dữ liệu truyền thống. Việc lưu trữ và khai thác lượng dữ liệu khổng lồ này để lọc ra được những dữ liệu hữu dụng là một thử thách lớn nhất mà chúng ta gặp phải trong xã hội hiện đại. Các hệ cơ sở dữ liệu quan hệ hiện tại bộc lộ những yếu kém, do đó trong những năm gần đây càng ngày càng có nhiều loại CSDL NoSQL được nghiên cứu và phát triển, những CSDL này đặc biệt thích hợp cho các ứng dụng cực lớn và nhỏ, giảm thiểu tối đa các phép tính toán, tác vụ đọc-ghi với khả năng chịu tải, chịu lỗi cao nhưng chỉ đòi hỏi về tài nguyên phần cứng thấp.

Trong đồ án này nhóm chúng em sẽ từng bước tìm hiểu CSDL NoSQL và tập trung vào tìm hiểu sâu hơn về hệ quản trị CSDL MongoDB.

Nhóm viết báo cáo này với mục đích giúp người sử dụng bước đầu tiếp cận, có cái nhìn khái quát về các CSDL hiện đại NoSQL, hiểu chi tiết hơn về hệ cơ sở dữ liệu cơ bản của NoSQL là MongoDB và đồng thời giúp người đọc có thể thực hiện một ứng dụng cơ bản trên hệ cơ sở dữ liệu MongoDB, với mong muốn tìm hiểu thêm công nghệ mới để áp dụng cho tương lai nghề nghiệp.

I. GIỚI THIỆU

Với hầu hết các thời kỳ web, Hệ quản trị cơ sở dữ liệu quan hệ dựa trên SQL đã thống trị hầu hết các hệ Quản trị Cơ sở dữ liệu. Tuy nhiên, thời gian gần đây, một cách tiếp cận mới đã bắt đầu biết đến là NoSQL, tạo ra sự thay thế cho các hệ quản trị cơ sở dữ liệu quan hệ truyền thống.

NoSQL còn có nghĩa là Non-Relational - không ràng buộc. Tuy nhiên, thuật ngữ đó ít phổ dụng hơn và ngày nay người ta thường dịch NoSQL thành Not Only SQL - Không chỉ SQL. NoSQL ám chỉ đến những cơ sở dữ liệu không dùng mô hình dữ liệu quan hệ để quản lý dữ liệu trong lĩnh vực phần mềm.

Thuật ngữ NoSQL được giới thiệu lần đầu vào năm 1998 sử dụng làm tên gọi chung cho các cơ sở dữ liệu quan hệ nguồn mở nhỏ nhưng không sử dụng SQL cho truy vấn.

Vào năm 2009, Eric Evans, nhân viên của Rackspace giới thiệu lại thuật ngữ NoSQL khi Johan Oskarsson của Last.fm muốn tổ chức một hội thảo về cơ sở dữ liệu nguồn mở phân tán. Thuật ngữ NoSQL đánh dấu bước phát triển của thế hệ CSDL mới: một thế hệ CSDL không ràng buộc, phân tán, nguồn mở, khả năng mở rộng theo chiều ngang, có thể lưu trữ, xử lý từ một lượng rất nhỏ cho tới hàng petabytes dữ liệu trong hệ thống có độ chịu tải, chịu lỗi cao với những đòi hỏi về tài nguyên phần cứng thấp.

Một số đặc điểm nhận dạng cho thế hệ CSDL mới này bao gồm: schema-free, hỗ trợ mở rộng dễ dàng, API đơn giản, nhất quán cuối (eventual consistency), không giới hạn không gian dữ liệu,...

Sau đây là danh sách các CSDL NoSQL:

- Wide Column Store / Column Families:

Hadoop/HBase – Apache, BigTable – Google, Cassandra - Facebook/Apache, Hypertable - Zvents Inc/Baidu, Cloudera, SciDB, Mnesia, Tablets,...

- Key-Value Store/Tuple store:

+ *Key/value cache in RAM:* memcached, Citrusleaf database, Velocity, Redis, Tuple space,...

+ *Key/value save on disk:* Memcachedb, Berkeley DB, Tokyo Cabinet, Redis,...

+ *Eventually Consistent Key Value Store:* Amazon Dynamo, Voldemort, Dynamite, KAI, Cassandra, Hibari, Project Voldemort,...

- + *Ordered key-value store*: NMDB, Memcachedb, Berkeley DB,...
- + *Distributed systems*: Apache River, MEMBASE, Azure Table Storage, Amazon Dynamo,...
- **Document Store**: Apache Jackrabbit, CouchDB, IBM Lotus Notes Storage Format (NSF), MongoDB, Terrastore, ThruDB, OrientDB, RavenDB,...
- **Graph Database**: Neo4J, Sones, AllegroGraph, Core Data, DEX, FlockDB, InfoGrid, OpenLink Virtuoso,...

Tuy cùng mang những đặc điểm chung của NoSQL nhưng mỗi CSDL NoSQL cũng có những đặc điểm riêng, và vì thế thường được dùng cho những dự án khác nhau. Ví dụ:

- MongoDB và Redis là những lựa chọn tốt cho việc lưu trữ các dữ liệu thống kê ít được đọc mà lại được viết thường xuyên.
- Hadoop, một CSDL dạng tự do, phân tán làm tốt công việc lưu trữ các dữ liệu lớn như các con số thống kê thời tiết hoặc công việc phân tích nghiệp vụ.
- Memcachedb, một CSDL nhất thời chóng tàn, tuyệt vời trong lưu trữ các phiên làm việc web, các khóa, và các con số thống kê ngắn hạn.
- Cassandra và Riak (các lưu trữ dư thừa, tự động tạo bó cluster) làm tốt trong các môi trường với các ứng dụng có tính sẵn sàng cao, khi thời gian sống tối đa là sống còn.

Để tìm hiểu sâu hơn về các CSDL hiện đại NoSQL, chúng ta đi nghiên cứu chi tiết CSDL đặc trưng là MongoDB.

1. MongoDB là gì:

MongoDB là một database hướng tài liệu (document), một dạng NoSQL database. Vì thế, MongoDB sẽ tránh cấu trúc table-based của relational database để thích ứng với các tài liệu như JSON có một schema rất linh hoạt gọi là BSON. **MongoDB** sử dụng lưu trữ dữ liệu dưới dạng Document JSON nên mỗi một collection sẽ các các kích cỡ và các document khác nhau. Các dữ liệu được lưu trữ trong document kiểu JSON nên truy vấn sẽ rất nhanh.

Trong những gương mặt góp phần làm suy tàn đế chế SQL thì MongoDB nổi lên là một CSDL đáng tin cậy và dễ dùng nhất. Mongo viết bằng C++. Nó thích hợp cho các ứng dụng tầm trung trở lên. Nếu tỉ lệ lượng dữ liệu ghi vào CSDL của ứng dụng lớn hơn lượng

đọc thì đây càng là lựa chọn hợp lý. MongoDB là một CSDL có khả năng mở rộng, hiệu suất cao, mã nguồn mở và hướng văn bản.



Hình 1. Biểu tượng của hệ quản trị cơ sở dữ liệu MongoDB

2. Lịch sử hình thành MongoDB

MongoDB được bắt đầu phát triển vào đầu năm 2007 khi công ty 10gen đang phát triển một nền tảng tương tự dịch vụ Azure của Microsoft. Công ty 10gen là một công ty phần mềm có trụ sở tại New York, nay được đổi tên thành MongoDB Inc. Việc phát triển ban đầu tập trung vào xây dựng [PaaS](#) (một nền tảng dịch vụ)

Năm 2009, MongoDB đã xuất hiện trên thị trường như một dự án mã nguồn mở máy chủ cơ sở dữ liệu và được duy trì bởi chính tổ chức này, được viết bởi ngôn ngữ C++

Tháng 3 năm 2010, MongoDB Inc. đã tung ra sản phẩm sẵn sàng đầu tiên của mình là phiên bản 1.4. Phiên bản ổn định tiếp theo của MongoDB là phiên bản 2.4.9 được phát hành vào ngày 10 tháng 1 năm 2014. Đầu năm 2015, phiên bản 3.0 được phát hành, cuối năm 2015 phiên 3.2 ra đời đi kèm với công cụ quản trị trên giao diện đồ họa [MongoDB Compass](#). Năm 2019, MongoDB Inc. đã hợp tác với Alibaba Cloud (công ty con thuộc tập đoàn Alibaba), cung cấp cho khách hàng giải pháp MongoDB dưới dạng dịch vụ.

MongoDB là một công ty toàn cầu có trụ sở chính của Hoa Kỳ tại New York city, USA và trụ sở quốc tế Dublin, Ireland. Vào ngày 30-10-2019, MongoDB đã hợp tác với Alibaba Cloud, công ty này sẽ cung cấp cho khách hàng của mình giải pháp MongoDB dưới dạng dịch vụ. Năm 2020, MongoDB được đánh giá là cơ sở dữ liệu NoSQL phổ biến nhất. Hiện nay, MongoDB nằm trong top 4 cơ sở dữ liệu được các nhà phát triển đánh giá cao trong khảo sát dành cho các nhà phát triển Stack Overflow. Với bộ tính năng đa điểm,

MongoDB được sử dụng bởi nhiều thương hiệu lớn như MTV Networks, Adobe, Google, Ebay, Facebook,...

3. Tác giả, tổ chức quản lý

MongoDB được thành lập vào năm 2007 bởi Dwight Merriman, Eliot Horowitz và Kevin Ryan – nhóm đứng sau DoubleClick. Tại công ty quảng cáo Internet DoubleClick (hiện thuộc sở hữu của Google), nhóm đã phát triển và sử dụng nhiều kho lưu trữ dữ liệu tùy chỉnh để khắc phục những thiếu sót của cơ sở dữ liệu hiện có. Doanh nghiệp phục vụ 400.000 quảng cáo mỗi giây, nhưng thường phải vật lộn với cả khả năng mở rộng và sự linh hoạt. Để giải quyết các vấn đề đó nhóm đã được truyền cảm hứng để tạo cơ sở dữ liệu giải quyết những thách thức mà họ gặp phải tại DoubleClick. Đây là lúc MongoDB ra đời.

MongoDB, Inc có trụ sở chính tại New York, với các văn phòng trên khắp Bắc Mỹ, Châu Âu và Châu Á - Thái Bình Dương và lực lượng lao động hiện chủ yếu làm việc từ xa, chúng tôi ở gần nơi bạn kinh doanh. MongoDB có hơn 39.100 khách hàng tại hơn 100 quốc gia. Nền tảng cơ sở dữ liệu MongoDB đã được tải xuống hơn 325 triệu lần và đã có hơn 1,5 triệu đăng ký cho các khóa học của Đại học MongoDB.

II. TỔNG QUAN VỀ MONGODB

Trước khi đi vào tìm hiểu kỹ hơn về MongoDB, chúng ta làm quen với một số khái niệm cơ bản của MongoDB:

- `_id` – Là trường bắt buộc có trong mỗi document. Trường `_id` đại diện cho một giá trị duy nhất trong document MongoDB. Trường `_id` cũng có thể được hiểu là khóa chính trong document. Nếu bạn thêm mới một document thì MongoDB sẽ tự động sinh ra một `_id` đại diện cho document đó và là duy nhất trong cơ sở dữ liệu MongoDB.
- Collection – Là nhóm của nhiều document trong MongoDB. Collection có thể được hiểu là một bảng tương ứng trong cơ sở dữ liệu RDBMS (Relational Database Management System). Collection nằm trong một cơ sở dữ liệu duy nhất. Các collection không phải định nghĩa các cột, các hàng hay kiểu dữ liệu trước.
- Cursor – Đây là một con trỏ đến tập kết quả của một truy vấn. Máy khách có thể lặp qua một con trỏ để lấy kết quả. Với kỹ thuật này một con trỏ được trả về. Con

trở sau đó được sử dụng lặp đi lặp lại để lấy tất cả các văn bản mà truy vấn trả về. Chúng ta có thể xem ví dụ sau:

```
> var cur = db.example.find();
> cur.forEach( function(x) { print(tojson(x))});
{"n" : 1 , "_id" : "497ce96f395f2f052a494fd4"}
{"n" : 2 , "_id" : "497ce971395f2f052a494fd5"}
{"n" : 3 , "_id" : "497ce973395f2f052a494fd6"}
>
```

- Database – Nơi chứa các Collection, giống với cơ sở dữ liệu RDMS chúng chứa các bảng. Mỗi Database có một tập tin riêng lưu trữ trên bộ nhớ vật lý. Một máy chủ MongoDB có thể chứa nhiều Database.
- Document – Một bản ghi thuộc một Collection thì được gọi là một Document. Các Document lần lượt bao gồm các trường tên và giá trị.
- Field – Là một cặp key – value trong một document. Một document có thể có không hoặc nhiều trường. Các trường giống các cột ở cơ sở dữ liệu quan hệ.
- JSON – Viết tắt của JavaScript Object Notation. Con người có thể đọc được ở định dạng văn bản đơn giản thể hiện cho các dữ liệu có cấu trúc. Hiện tại JSON đang hỗ trợ rất nhiều ngôn ngữ lập trình.
- Index – Là những cấu trúc dữ liệu đặc biệt, dùng để chứa một phần nhỏ của các tập dữ liệu một cách dễ dàng để quét. Chỉ số lưu trữ giá trị của một fields cụ thể hoặc thiết lập các fields, sắp xếp theo giá trị của các fields này. Index hỗ trợ độ phân tích một cách hiệu quả các truy vấn. Nếu không có chỉ mục, MongoDB sẽ phải quét tất cả các documents của collection để chọn ra những document phù hợp với câu truy vấn. Quá trình quét này là không hiệu quả và yêu cầu MongoDB để xử lý một khối lượng lớn dữ liệu.

Hãy lưu ý sự khác biệt của các trường và _id trong một document. Một _id được dùng để đại diện cho một document và chúng được sinh ra khi thêm một Document vào Collection.

RDBMS	MongoDB
Database	Database
Table	Collection
Tuple/Row	Document
column	Field
Table Join	Embedded Documents
Primary Key	Primary Key (Default key _id provided by mongodb itself)
Database Server and Client	
Mysqld/Oracle	mongod
mysql/sqlplus	mongo

Hình 2. So sánh khái niệm giữa MongoDB và SQL

1. Các kiểu dữ liệu trong MongoDB

- **Chuỗi:** Đây là kiểu dữ liệu được sử dụng phổ biến nhất để lưu giữ dữ liệu. Chuỗi trong MongoDB phải là UTF-8 hợp lệ.
- **Số nguyên:** Kiểu dữ liệu này được sử dụng để lưu một giá trị số. Số nguyên có thể là 32 bit hoặc 64 bit phụ thuộc vào Server của bạn.
- **Boolean:** Kiểu dữ liệu này được sử dụng để lưu giữ một giá trị Boolean (true/false).
- **Double:** Kiểu dữ liệu này được sử dụng để lưu các giá trị số thực dấu chấm động.
- **Min/ Max keys:** Kiểu dữ liệu này được sử dụng để so sánh một giá trị với các phần tử BSON thấp nhất và cao nhất.
- **Mảng:** Kiểu dữ liệu này được sử dụng để lưu giữ các mảng hoặc danh sách hoặc nhiều giá trị vào trong một key.
- **Timestamp:** Đánh dấu thời điểm một Document được sửa đổi hoặc được thêm vào.
- **Object:** Kiểu dữ liệu này được sử dụng cho các Document được nhúng vào.
- **Null:** Kiểu dữ liệu này được sử dụng để lưu một giá trị Null.
- **Symbol:** Kiểu dữ liệu này được sử dụng giống như một chuỗi
- **Date :** Kiểu dữ liệu này được sử dụng để lưu giữ date và time hiện tại trong định dạng UNIX time.
- **Object ID:** Kiểu dữ liệu này được sử dụng để lưu giữ ID của Document.
- **Binary data:** Kiểu dữ liệu này được sử dụng để lưu giữ dữ liệu nhị phân.

- **Code:** Kiểu dữ liệu này được sử dụng để lưu giữ JavaScript code vào trong Document.
- **Regular expression:** Kiểu dữ liệu này được sử dụng để lưu giữ Regular Expresion.

Bảng 1. Các kiểu dữ liệu trong MongoDB

Type	Number	Alias
Double	1	“double”
String	2	“string”
Object	3	“object”
Array	4	“array”
Binary data	5	“binData”
Undefined	6	“undefined”
ObjectId	7	“objectId”
Boolean	8	“bool”
Date	9	“date”
Null	10	“null”
Regular Expression	11	“regex”
DBPointer	12	“dbPointer”
JavaScript	13	“javascript”
Symbol	14	“symbol”
JavaScript (with scope)	15	“javascriptWithScope”
32-bit integer	16	“int”
Timestamp	17	“timestamp”
64-bit integer	18	“long”
Decimal128	19	“decimal”
Min key	-1	“minKey”
Max key	127	“maxKey”

2. Một số câu lệnh dùng trong MongoDB

Để trực quan, chúng ta so sánh các câu lệnh thường dùng trong MongoDB với SQL

Bảng 2. Một số câu lệnh dùng trong MongoDB

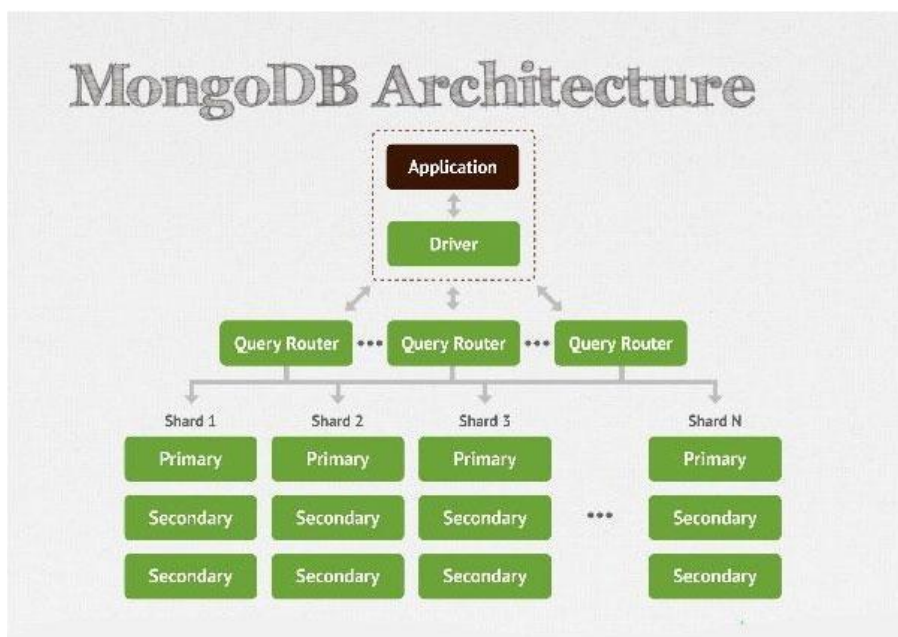
Câu lệnh	SQL	MongoDB
Create table	CREATE TABLE people (id MEDIUMINT NOT NULL AUTO_INCREMENT, user_id Varchar(10), age Number, status char(2), PRIMARY KEY (id))	db.createCollection('people') hoặc db.people.insertOne({User_id: "abc123", Age: 20, Status: "DZ"})
Drop table	DROP TABLE people	db.people.drop()
Insert records into tables	INSERT INTO people(user_id, age, status) VALUES ("bcd456", 19, "XG")	db.people.insertOne({ user_id: "bcd456", age: 19, status: "XG" })
Select	SELECT *FROM people	db.people.find()
	SELECT id,user_id, status FROM people	db.people.find({ }, { user_id: 1, status: 1 })
	SELECT * FROM people WHERE status = "DZ" AND age = 20	db.people.find({ status: "DZ", age: 20 })
	SELECT * FROM people WHERE status = "XG" OR age = 19	db.people.find({ \$or: [{ status: "XG" } , { age: 19 }] })
	SELECT * FROM people WHERE user_id like "%bc%"	db.people.find({ user_id: /bc/ }) db.people.find({ user_id: { \$regex: /bc/ } })
	SELECT COUNT(user_id) FROM people	db.people.count({ user_id: { \$exists: true } }) db.people.find({ user_id: { \$exists: true } }).count()

Update records	<code>UPDATE people SET status = "BD" WHERE age > 30</code>	<code>db.people.updateMany({ age: { \$gt: 30 } }, { \$set: { status: "BD" } })</code>
	<code>UPDATE people SET age = age + 1 WHERE status = "XG"</code>	<code>db.people.updateMany({ status: "XG" } , { \$inc: { age: 1 } })</code>
Delete Records	<code>DELETE FROM people WHERE status = "BD"</code>	<code>db.people.deleteMany({ status: "BD" })</code>
	<code>DELETE FROM people</code>	<code>db.people.deleteMany({})</code>

3. MongoDB hoạt động như thế nào?

- MongoDB hoạt động dưới một tiến trình ngầm service, luôn mở một cổng (Cổng mặc định là 27017) để lắng nghe các yêu cầu truy vấn, thao tác từ các ứng dụng gửi vào sau đó mới tiến hành xử lý.

- Mỗi một bản ghi của MongoDB được tự động gắn thêm một field có tên “_id” thuộc kiểu dữ liệu ObjectId mà nó quy định để xác định được tính duy nhất của bản ghi này so với bản ghi khác, cũng như phục vụ các thao tác tìm kiếm và truy vấn thông tin về sau. Trường dữ liệu “_id” luôn được tự động đánh index (chỉ mục) để tốc độ truy vấn thông tin đạt hiệu suất cao nhất.



Hình 3. Quy trình hoạt động của MongoDB

- Mỗi khi có một truy vấn dữ liệu, bản ghi được cache (ghi đệm) lên bộ nhớ Ram, để phục vụ lượt truy vấn sau diễn ra nhanh hơn mà không cần phải đọc từ ổ cứng.

- Khi có yêu cầu thêm/sửa/xóa bản ghi, để đảm bảo hiệu suất của ứng dụng mặc định MongoDB sẽ chưa cập nhật xuống ổ cứng ngay, mà sau 60 giây MongoDB mới thực hiện ghi toàn bộ dữ liệu thay đổi từ RAM xuống ổ cứng.

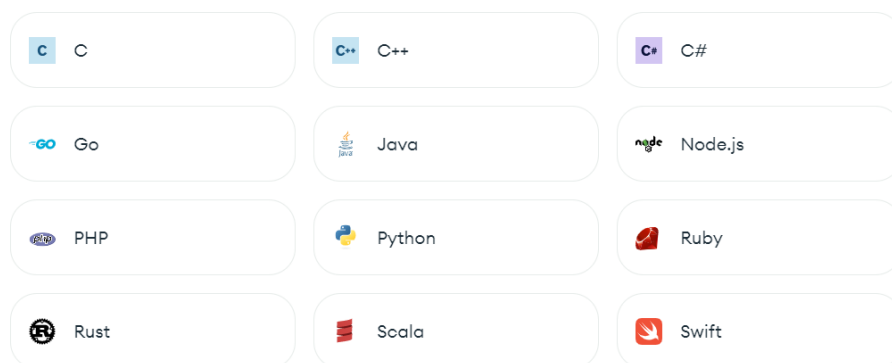
4. Ngôn ngữ thao tác với dữ liệu

Các document của MongoDB sử dụng format JSON (JavaScript Object Notation), đây là một chuẩn lưu trữ, trao đổi dữ liệu đơn giản và gọn nhẹ. Với ưu điểm dễ đọc, dễ hiểu, đa phần các ngôn ngữ lập trình phổ biến hiện nay đều hỗ trợ JSON như: C, C++, C#, Java, JavaScript, Perl, Python,....

Start Developing with MongoDB

Connect your application to your database with one of our official libraries.

The following libraries are officially supported by MongoDB. They are actively maintained, support new MongoDB features, and receive bug fixes, performance enhancements, and security patches.



Hình 4. MongoDB Drivers

5. Mô hình lưu trữ

5.1. Văn bản (documents)

Văn bản là một khái niệm quan trọng trong MongoDB. Văn bản bao gồm tập hợp các khóa với các giá trị tương ứng.

Ví dụ: `{"greeting" : "Hello, world!"}`

Văn bản trên gồm một khóa là “greeting”, với giá trị là “Hello, world!”. Các văn bản có thể chứa nhiều cặp khóa/giá trị.

Ví dụ: `{"greeting" : "Hello, world!", "foo" : 3}`

* Một số lưu ý:

- Các cặp khóa/ giá trị trong văn bản được sắp xếp. Văn bản trên sẽ khác với văn bản sau:

`{"foo" : 3, "greeting" : "Hello, world!"}`

- Khóa trong văn bản là một chuỗi
- MongoDB phân biệt chữ hoa chữ thường
- Văn bản trong MongoDB không được chứa những khóa giống nhau. Ví dụ văn bản sau là không hợp lệ

```
{"greeting" : "Hello, world!", "greeting" : "Hello, MongoDB!"}
```

5.2. Bộ sưu tập (collections)

Bộ sưu tập là một nhóm các văn bản. Nếu văn bản tương đương với dòng trong CSDL quan hệ thì bộ sưu tập tương đương với bảng.

Bộ sưu tập là một Schema-Free, nghĩa là các văn bản có hình dạng khác nhau có thể cùng được lưu trữ trong 1 bộ sưu tập.

Ví dụ các văn bản sau có thể cùng được lưu trong một bộ sưu tập:

```
{"greeting" : "Hello, world!"}
```

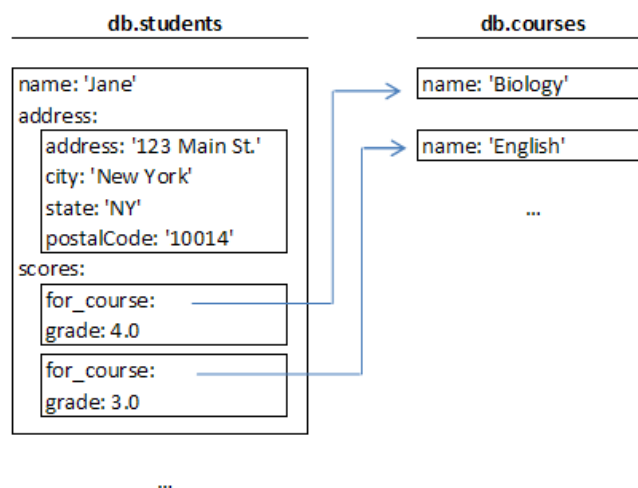
```
{"foo" : 5}
```

Bộ sưu tập được xác định bởi tên của nó là một chuỗi UTF-8

5.3. Thiết kế lược đồ

Với MongoDB, chúng ta ít phải “chuẩn hóa” hơn so với khi làm việc với lược đồ quan hệ vì trong MongoDB không có khái niệm liên kết (join). Nói chung, với mỗi đối tượng (object) mức cao nhất, ta sẽ có một bộ sưu tập (collection) dữ liệu.

Một bộ sưu tập không phải cho tất cả các lớp (class), thay vào đó, các đối tượng sẽ được nhúng vào đó.



Hình 5. Minh họa bộ sưu tập

Hình 5 minh họa có 2 bộ sưu tập: students và courses. Các văn bản student được nhúng văn bản address và văn bản score. Trong đó, văn bản Score được tham chiếu đến Courses.

So sánh với lược đồ quan hệ: ta cần lưu Score vào bảng riêng và dùng khóa ngoài liên kết với Student.

Lựa chọn chỉ mục

Một khía cạnh thứ hai khi thiết kế lược đồ là việc lựa chọn chỉ mục. Việc đánh chỉ mục làm cho việc thực hiện truy vấn nhanh hơn. Một truy vấn bình thường cần vài phút, có thể được thực hiện ngay lập tức với việc sử dụng chỉ mục.

Trong MongoDB:

- Trường `_id` được đánh chỉ mục tự động
- Những trường mà theo đó các khóa được tìm kiếm nên được đánh chỉ mục
- Những trường sắp xếp nói chung nên được đánh chỉ mục

Lưu ý rằng việc thêm vào chỉ mục chỉ làm chậm quá trình ghi vào bộ sưu tập mà không làm chậm quá trình đọc. Vì vậy, sử dụng nhiều chỉ mục với những bộ sưu tập mà tỉ lệ read:write cao. Với những bộ sưu tập mà ghi nhiều hơn đọc, sử dụng chỉ mục là rất tốn kém.

5.4. Chỉ mục (Index)

Chỉ mục làm tăng hiệu suất truy vấn lên rất nhiều. Điều quan trọng là nghĩ xem xét tất cả các loại truy vấn cần trong ứng dụng để xác định những chỉ mục liên quan. Khi đã xác định xong, việc tạo ra các chỉ mục trong MongoDB là khá dễ dàng.

5.4.1. Các khái niệm cơ bản

Chỉ mục là một cấu trúc dữ liệu, thu thập thông tin về giá trị của các trường trong các văn bản của một bộ sưu tập. Cấu trúc dữ liệu này được sử dụng trong tối ưu truy vấn Mongo để sắp xếp nhanh các văn bản trong một bộ sưu tập.

Chúng ta có thể khởi tạo chỉ mục bằng cách gọi hàm `ensureIndex()` và cung cấp một văn bản với một hoặc nhiều khóa để đánh chỉ mục. Ví dụ đánh chỉ mục cho trường `name` trong `students`:

```
db.students.ensureIndex({name:1});
```

Hàm `ensureIndex()` chỉ khởi tạo chỉ mục nếu nó chưa tồn tại. Để kiểm tra việc tồn tại chỉ mục trên bộ sưu tập `students`, ta có thể chạy hàm `db.students.getIndexes()`.

Khi một bộ sưu tập được đánh chỉ mục trên một khóa nào đó, truy cập ngẫu nhiên trên biểu thức truy vấn có chứa khóa đó sẽ được thực hiện rất nhanh. Nếu không được đánh chỉ mục, MongoDB phải soát tất cả các văn bản để kiểm tra giá trị của khóa đó trong truy vấn.

5.4.2. Chỉ mục mặc định

Một chỉ mục luôn luôn được tạo ra là `_id`. Chỉ mục này là đặc biệt và không thể bị xóa. Chỉ mục `_id` là duy nhất cho các khóa của nó.

5.4.3. Các khóa nhúng

Với MongoDB chúng ta thậm chí có thể đánh chỉ mục trên các khóa bên trong văn bản nhúng. Ví dụ: `db.students.ensureIndex({"address.city": 1})`

5.4.4. Xóa chỉ mục

Xóa tất cả các chỉ mục trên bộ sưu tập:

```
db.collection.dropIndexes();
```

Xóa chỉ mục đơn:

```
db.collection.dropIndex({x: 1, y: -1})
```

6. Nhân bản (Replication)

Có lẽ công việc quan trọng nhất của bất kỳ quản trị viên MongoDB là đảm bảo sao cho sao chép được thiết lập và hoạt động đúng. Sao chép có thể được sử dụng hoàn toàn để dự phòng và toàn vẹn dữ liệu hoặc có thể được sử dụng cho mục đích cao hơn như mở rộng đọc, sao lưu nóng,...

MongoDB hỗ trợ sao chép dữ liệu không đồng bộ giữa các máy chủ. Tại một thời điểm, chỉ có 1 máy chủ hoạt động để ghi (primary hay master).

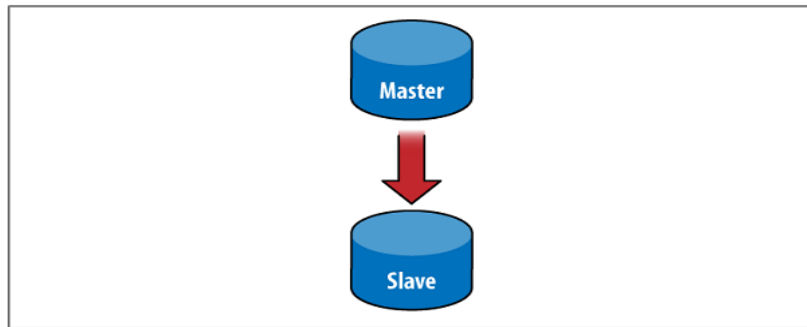
Có hai hình thức sao chép.

- Master-Slave Replication
- Replica Sets.

6.1. Master-Slave Replication

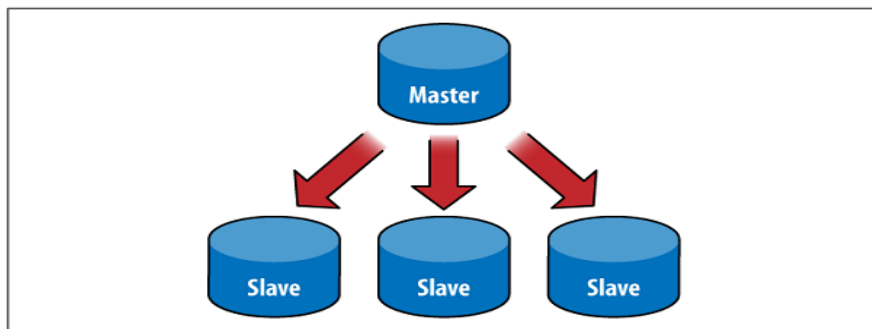
Sao chép Master-slave là mô hình sao chép phổ biến nhất được hỗ trợ bởi MongoDB. Mô hình này rất linh hoạt và có thể được sử dụng để sao lưu, dự phòng, mở rộng đọc, ...

Hình 6 minh họa mô hình Master – Slave bao gồm 2 nút, một nút làm Master, nút còn lại làm Slave



Hình 6. Mô hình Master – Slave hai nút

Hình 7 minh họa mô hình Master – Slave bao gồm 4 nút, một nút làm Master, 3 nút còn lại làm Slave



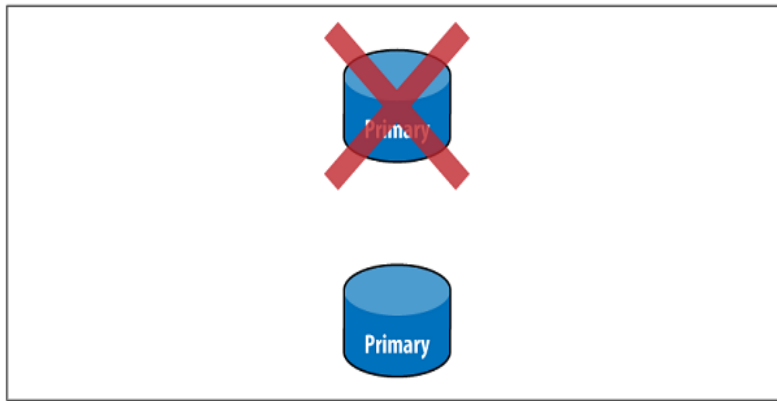
Hình 7. Mô hình Master – Slave bốn nút

Để thiết lập cần khởi động nút master và một hoặc nhiều nút slave, các nút này đều biết địa chỉ của nút master. Để khởi động master, chạy `mongod --master`. Để khởi động slave, chạy `mongod --slave --source master_address`, trong đó `master_address` là địa chỉ của nút master vừa được khởi động

6.2. Replica Sets

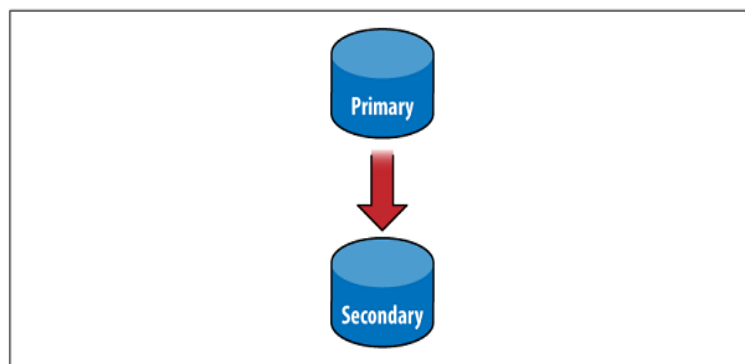
Replica Sets là một cụm master-slave tự động chịu lỗi. Replica Sets không có một master cố định: một master được bầu chọn và có thể thay đổi đến nút khác nếu master bị sập [1].

Hình 8 mô phỏng mô hình Replica Sets gồm 2 nút.



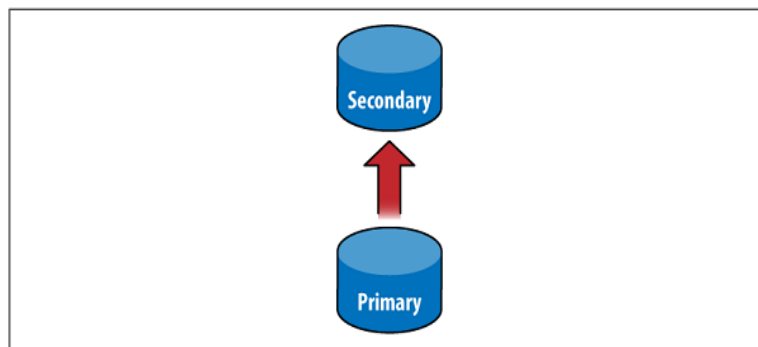
Hình 8. Mô hình Replica Sets hai nút

Khi server chính chết, server cấp 2 chờ thành server chính (hình 9).



Hình 9. Replica Sets – Bầu chọn master mới

Nếu server chính ban đầu hoạt động trở lại, nó trở thành server cấp 2 (hình 10).



Hình 10. Server chính trở thành server cấp 2

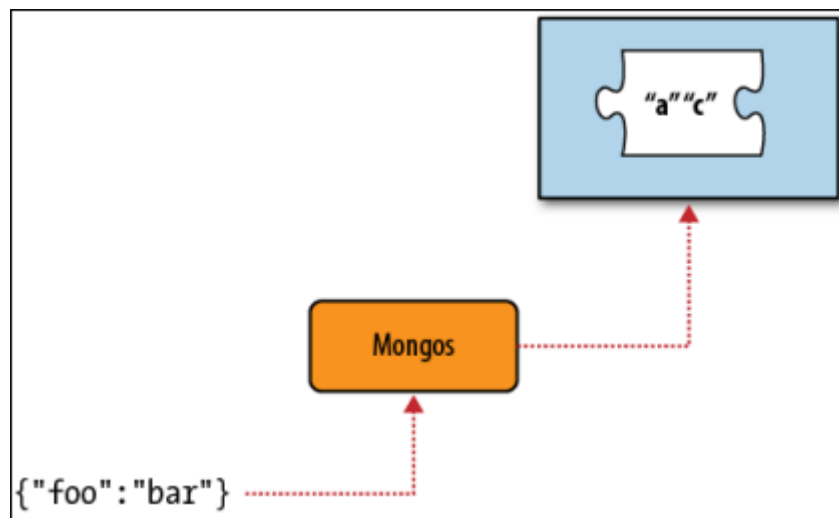
7. Cơ chế phân tán

7.1. Sharding

Sharding là cơ chế tự động của MongoDB dùng để chia tách một dữ liệu kích thước lớn cho rất nhiều server (thường gọi là cluster). Sharding được thiết kế để phục vụ 3 mục điều cơ bản sau:

7.1.1. Làm cho cluster “trong suốt” với người dùng:

Để hoàn thành nhiệm vụ này, MongoDB sử dụng một quá trình routing đặc biệt gọi là mongos. Mongos đứng trước cluster, đóng vai trò điều phối việc truy cập đến shard nào, trả dữ liệu từ shard nào ra. Nó chuyển tiếp các request tới các server khác có tài nguyên hoặc đến cluster đằng sau nó. Sau đó lắp ráp lại, và gửi các response lại về cho các client. Do đó, các client không cần biết rằng chúng đang giao tiếp với cluster nào thật sự mà chỉ biết rằng mình đang kết nối tới một server bình thường. Đây gọi là tính “trong suốt” với người sử dụng.



Hình 11. Cluster “trong suốt” với người dùng

7.1.2. Làm cho cluster luôn sẵn sàng để đọc hoặc ghi:

Một cluster còn tồn tại phải đảm bảo được rằng nó luôn sẵn sàng. Mỗi phần con trong cluster sẽ có ít nhất một vài tiến trình phục vụ dự bị trên máy khác.

7.1.3. Làm cho cluster phát triển “tự nhiên”

Bất cứ khi nào người dùng cần thêm dung lượng, họ có thể thêm một cách dễ dàng. Mỗi cluster khi được quản lý lại “thể hiện” như một node riêng lẻ và dễ dàng config.

Cơ sở dữ liệu với lượng dữ liệu lớn hoặc tải cao là một thách thức lớn đối với mô hình máy chủ đơn (Mainframe).

Với các câu lệnh truy vấn phức tạp có thể làm CPU tăng cao. Với việc phải làm việc với dữ liệu lớn có thể đòi hỏi lượng RAM của máy chủ.

Hiện nay có 2 phương án có thể giải quyết vấn đề trên là: Vertical Scaling và Horizontal Scaling

- Vertical Scaling (Scaling Up): tăng số lõi CPU hoặc là dùng CPU mạnh hơn
- Horizontal Scaling (Scaling Out) : tăng số máy chủ.

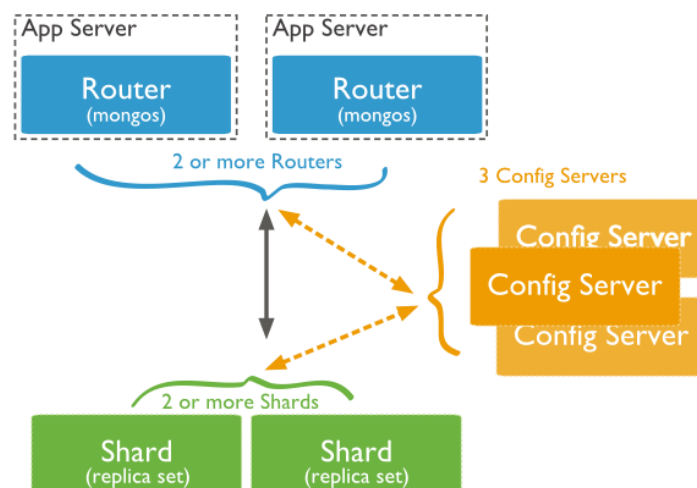
Sharding là một phương pháp để lưu trữ dữ liệu (storage) của DB trên nhiều máy chủ. MongoDB sử dụng sharding để hỗ trợ việc phân tán một lượng dữ liệu trên nhiều máy chủ, ở đây có thể là các tập collection trong DB điều này giúp cho việc truy cập nhanh hơn, giảm tải việc quá tải ổ cứng cho một vài máy chủ và giúp hệ thống dễ dàng mở rộng khi có nhu cầu hơn.

Ví dụ, nếu một cơ sở dữ liệu có một bộ dữ liệu 1 terabyte, và có chia ra thành 4 shards, sau đó mỗi shard có thể giữ chỉ 256GB dữ liệu. Nếu có 40 shard thì mỗi shard có thể giữ chỉ 25GB dữ liệu như hình dưới. Điều này giảm dữ liệu mỗi máy chủ phải chứa và dễ dàng có thể mở rộng hệ thống theo chiều ngang.

* Sharded Cluster

Mongodb Sharded Cluster bao gồm các thành phần chính như sau:

- *Shards*: Là nơi chứa dữ liệu, được phân tán bởi nhiều máy chủ theo cơ chế “replica set”
- *Query routers*: Là nơi điều hướng việc client truy cập chính xác dữ liệu vào shard nào, mỗi hệ thống Sharding có thể có nhiều query router.
- *Config servers*: Là nơi chứa các “meta data” (nôm na là các thông số kỹ thuật) hệ thống Sharding, nó chứa một bản đồ dữ liệu của việc thiết lập các Shards. Query routers dùng các “meta data” này xác định được chính xác việc truy vấn vào Shards nào trên hệ thống. Mỗi hệ thống Sharding có chính xác là 3 file Config servers.



Hình 12. Sharded Cluster

Trong sơ đồ trên, có ba thành phần chính:

- **Shards:** được sử dụng để lưu giữ dữ liệu. Chúng cung cấp tính khả dụng cao và dữ liệu có tính đồng nhất. Trong môi trường tạo lập, mỗi Shard là một Replica Set riêng biệt.
- **Config Servers:** lưu giữ metadata của Cluster. Dữ liệu này chứa một ánh xạ của tập dữ liệu của Cluster tới Shards. Query Router sử dụng metadata này để hướng các hoạt động tới Shards cụ thể. Trong môi trường tạo lập, sharded clusters có chính xác 3 Config Servers.
- **Query Routers:** về cơ bản nó là mongo instance, giao diện với Ứng dụng Client và hướng các hoạt động tới Shard phù hợp. Query Router xử lý và hướng các hoạt động tới Shard và sau đó trả kết quả về Clients. Một Sharded Cluster có thể chứa nhiều hơn một Query Router để phân chia việc tải yêu cầu từ Client. Một Client gửi các yêu cầu tới một Query Router. Nói chung, một Sharded Cluster có nhiều Query Routers.

Mongodb chia sẻ dữ liệu ở tầng Collection (Tương ứng với Table trong hệ quản trị cơ sở dữ liệu quan hệ). Dữ liệu trong Collection được phân tán trên các Shards trong cụm.

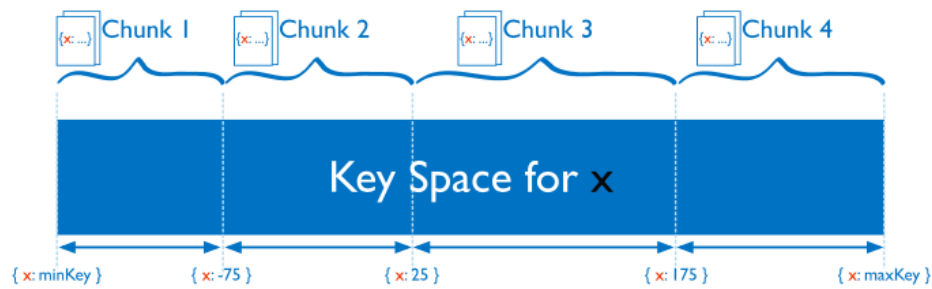
- **Shard Keys:** Để phân tán dữ liệu của Collection trên các Shards, MongoDB phân vùng dữ liệu thông qua Shard Keys. Shard Key là một cột hoặc một cụm các cột không thể thay đổi tồn tại ở mỗi Documents thuộc Collection. Chúng ta cần phải chọn một Shard Keys mỗi khi chúng ta phân chia dữ liệu của Collection.

Tất cả các Shard Collection bắt buộc phải tạo Index cho Shard Keys. Index này có thể dựa trên field hoặc một tập hợp các fields của Key. Đối với các Collection không có dữ liệu, MongoDB sẽ tự tạo index trên Shard Key nếu như Index không tồn tại.

* *Lưu ý:* Việc chọn lựa Shard Key có ảnh hưởng đến hiệu năng, khả năng mở rộng của Cluster.

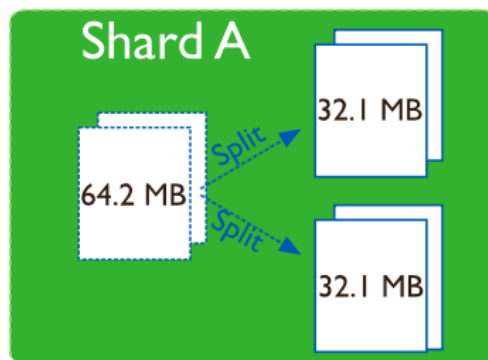
MongoDb hỗ trợ Database tồn tại cả 2 loại Sharding Collection và Non Sharding Collection.

Chunk: MongoDb phân chia dữ liệu Sharding Collection vào trong chunk. Mặc định Chunk Size là 64Mb.



Hình 13. Chunk

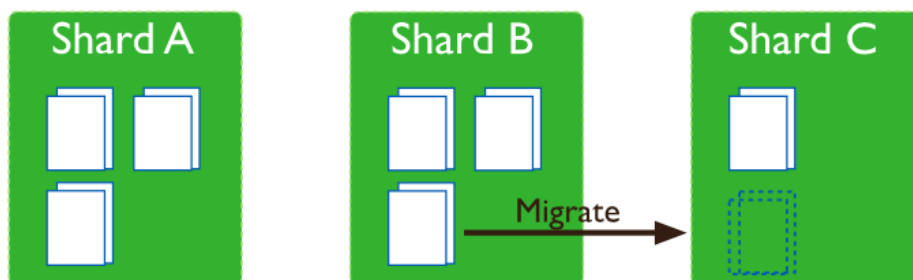
Chunk Splits: Là tiến trình chia cắt khi Chunk tăng đến giới hạn(Chunk Size) hoặc khi số lượng Document trong Chunk vượt quá tham số Maximum Number of Documents Per Chunk to Migrate. Dựa trên Share Key, MongoDB sẽ tiến hành chia cắt thành nhiều Chunk nếu cần.



Hình 14. Chunk Splits

Chunk Migration: MongoDB chuyển đổi Chunk để phân bố lại chunk của Shard Collection. Hình dung việc này cũng như việc chúng ta Defragment lại ổ cứng trên Window. MongoDB cũng hỗ trợ 2 phương pháp để làm được việc này:

- *Manual:* Chỉ nên sử dụng trong một số trường hợp nhất định như phân tán dữ liệu sử dụng Bulk Insert
- *Automatic:* Thông qua tiến trình Balancer tự động Migrate lại chunk khi có sự phân bố không đều giữa các chunk trên các Shards



Hình 15. Chunk Migration

7.1.4. Lợi ích của việc sử dụng Sharding

- Đọc/Ghi: MongoDB phân tán dữ liệu trên các Shards của Cluster. Việc đọc ghi sử dụng Sharding có thể dễ dàng được scaled horizontally
- Lưu trữ(Storage Capacity): Shardings phân tán dữ liệu trên các Shard của Cluster. Việc này giúp cho dữ liệu sẽ được phân bố đều trên cụm máy cài Sharding
- Tính sẵn sàng(High Availability): MongoDB có thể đọc ghi dữ liệu dễ dàng thậm chí ngay cả khi một hoặc nhiều Shard không sẵn sàng. Khi một hoặc nhiều Shard không sẵn sàng, MongoDB vẫn dễ dàng đọc ghi trên nhưng Share sẵn sàng

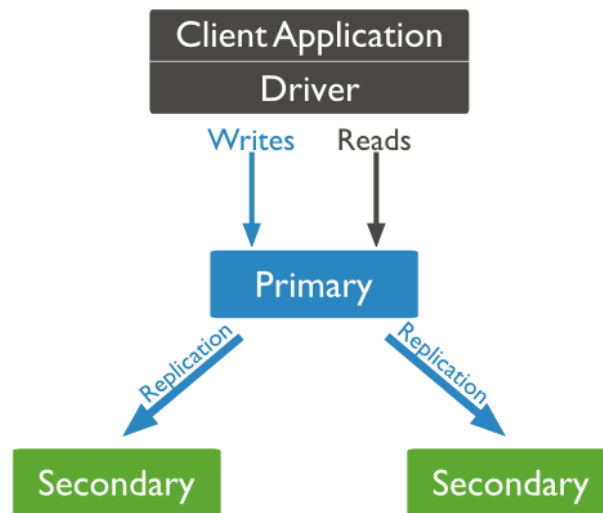
7.2. Replication

Replication là tiến trình đồng bộ hóa dữ liệu từ nhiều Server. Replication cung cấp sự dư thừa và tăng dữ liệu có tính khả dụng với nhiều bản sao dữ liệu trên nhiều Database Server khác nhau. Replication bảo vệ một cơ sở dữ liệu từ việc thất thoát của một Server nào đó. Replication cũng cho phép bạn phục hồi dữ liệu từ việc các lỗi ở phần cứng hoặc từ việc ngắt kết nối dịch vụ. Với các bản sao dữ liệu bổ sung, bạn có thể sử dụng cho việc phục hồi, báo cáo, hoặc backup.

MongoDB sử dụng Replica Set để thực hiện Replication. Một Replica Set là một nhóm các sự thể hiện của mongodb mà host cùng tập hợp dữ liệu đó. Trong một Replica, một node là Primary node (có thể gọi là node thứ cấp) sẽ nhận tất cả các hoạt động ghi. Tất cả sự thể hiện (instance) khác, thứ cấp, áp dụng các hoạt động từ node thứ cấp để mà chúng có cùng tập dữ liệu. Replica Set có thể chỉ có một node thứ cấp.

- Replica Set là một nhóm của hai hoặc nhiều node (nói chung, cần tối thiểu 3 node).
- Trong một Replica Set, một node là node thứ cấp và các node còn lại là sơ cấp.
- Tất cả dữ liệu tái tạo từ node sơ cấp đến node thứ cấp.
- Tại thời điểm duy trì tự động, việc lựa chọn thiết lập cho sơ cấp và một node sơ cấp được lựa chọn.
- Sau khi phục hồi node đã thất bại, một lần nữa nó lại kết hợp Replica Set và làm việc như một node thứ cấp.

Dưới đây là một sơ đồ đặc trưng cho Replication trong MongoDB, trong đó ứng dụng ở Client luôn luôn tương tác với node sơ cấp và node sơ cấp này sau đó tái tạo dữ liệu cho node thứ cấp.

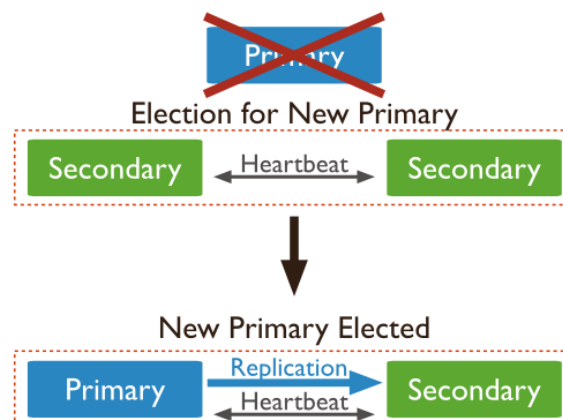


Hình 16. Cơ chế hoạt động Replica Set Member

* Đặc điểm của Replica Set:

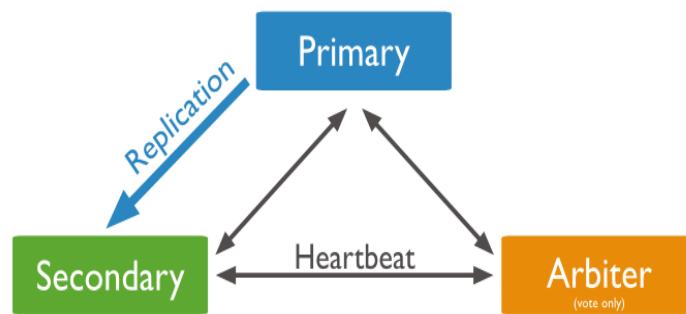
- Một Cluster gồm N node
- Bất kỳ node nào có thể là sơ cấp
- Tất cả hoạt động ghi là ở sơ cấp
- Tự động duy trì
- Tự động phục hồi

Một replica set chỉ có duy nhất một primary. Primary member sẽ thực hiện toàn bộ thao tác lên Database. Primary ghi các thay đổi của nó vào oplog - một file có vai trò như ghi lại lịch sử tác động (thay đổi) trên MySQL. Các secondary đã kết nối và apply từ primary nên sẽ được phân tán dataset với primary. Một replica set có thể có tối đa là 50 member. Nếu lớn hơn 50 member thì chúng ta phải dùng giải pháp khác.



Hình 17. Cơ chế Replica Set Member khi có sự cố

Cơ chế replica set khắc phục sự cố bằng thể thức thực hiện cuộc bầu cử - voting. Một secondary sẽ được bầu lên làm primary của cả hệ thống. Để giảm chi phí đầu tư, chúng ta có thể chỉ cần hai member trong một replica set. Thành viên thứ ba sẽ là một arbiter. Arbiter là một member đặc biệt. Nó không apply các oplog từ primary nên nó không lưu data gì cả. Nhiệm vụ của arbiter là giám sát hệ replica set qua các đường liên kết heartbeat và bầu chọn một secondary lên primary khi có sự cố xảy ra. Arbiter hoạt động không có gì nặng nề, bạn không cần server riêng cho arbiter nhưng arbiter không nên được cấp quyền trên server dùng làm primary hay secondary trong replica set. Arbiter thì sẽ mãi là arbiter không như primary có thể từ chức rồi trở thành secondary khi trở lại lại vào replica set hay secondary có thể được bầu trở thành primary khi sự cố xảy ra.



Hình 18. Cơ chế Replica Set Member có Arbiter

III. HƯỚNG DẪN CÀI ĐẶT

1. Cài đặt trên một máy

MongoDB đều hỗ trợ cho rất nhiều phiên bản của các hệ điều hành khác nhau. Có thể tham khảo chi tiết thêm tại đường link:

<https://www.mongodb.com/docs/manual/installation/#supported-platforms>.

Chúng tôi sẽ hướng dẫn cài đặt trên hệ điều hành Windows.

1.1. Chi tiết cấu hình máy

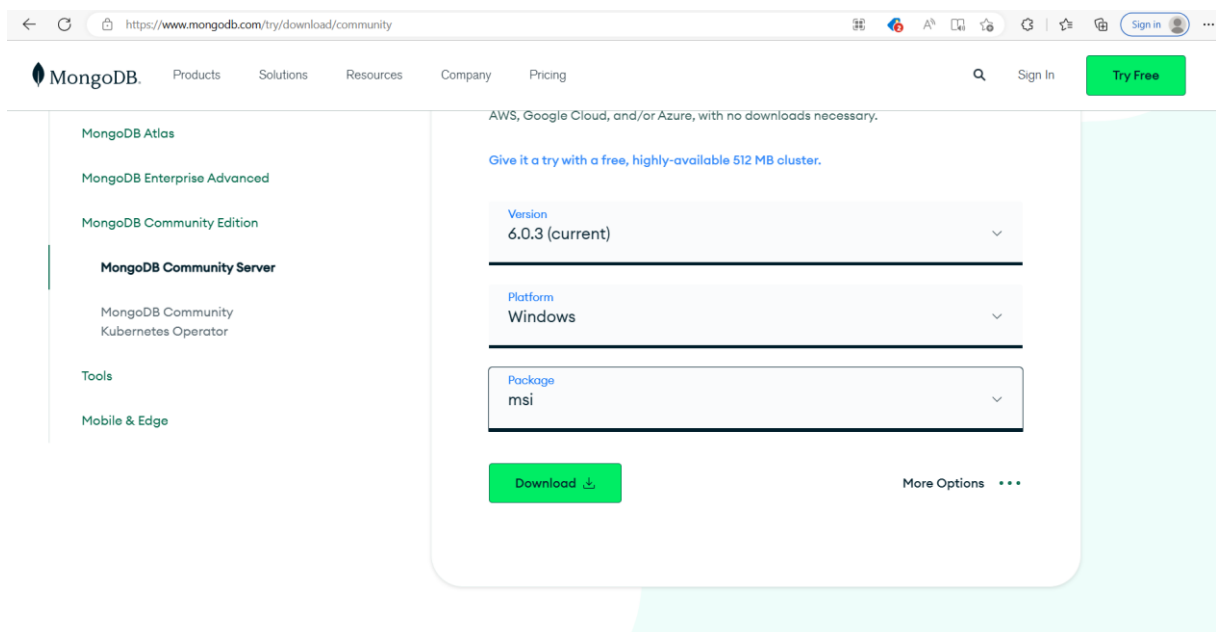
- Phiên bản hệ điều hành tối thiểu: Window 7
- Ram: lớn hơn 1 GB (MongoDB yêu cầu khoảng 1 GB RAM trên 100.000 assets, nhu cầu sử dụng sẽ quyết định đến dung lượng Ram tối thiểu cần đáp ứng)

1.2. Các bước thực hiện

Chúng tôi sẽ hướng dẫn cài đặt phiên bản MongoDB Community Server

- **Bước 1:** Truy cập vào trang chủ <https://www.mongodb.com/home> → chọn *Products* → chọn *Community Server* → *MongoDB Community Server*

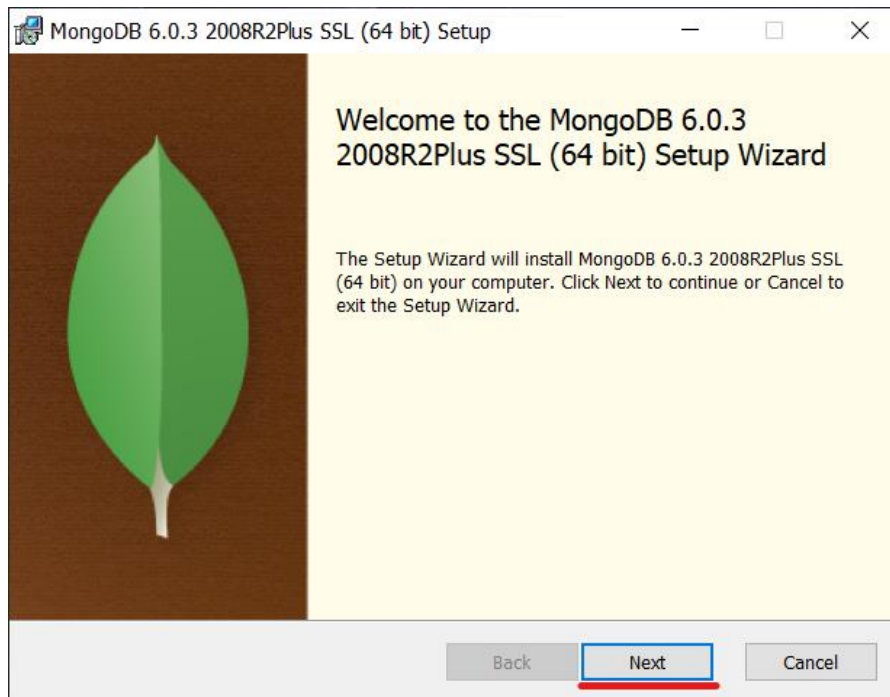
hoặc truy cập link: <https://www.mongodb.com/try/download/community>



Hình 19. Trang chủ tải MongoDB

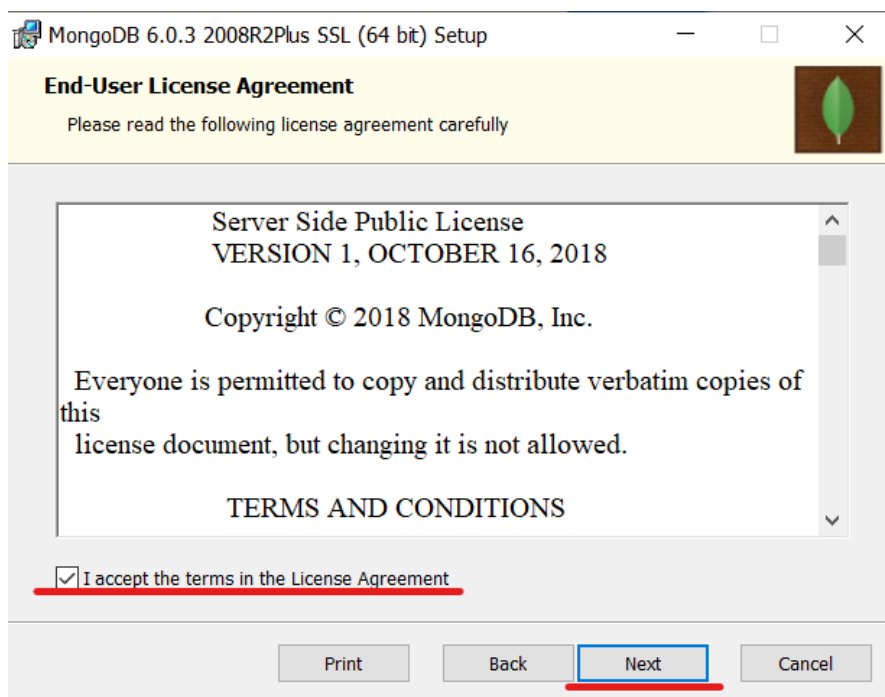
Tại đây chọn phiên bản MongoDB và hệ điều hành cần thiết, ở mục Package nên chọn tập *msi* để tối ưu hóa việc cài đặt. Ấn *Download* để tải trình cài đặt về

- **Bước 2:** Mở file vừa download về, ấn Next



Hình 20. Cài đặt MongoDB

- **Bước 3:** Tích vào ô **I accept the terms in the License Agreement** => sau đó bấm vào **Next**

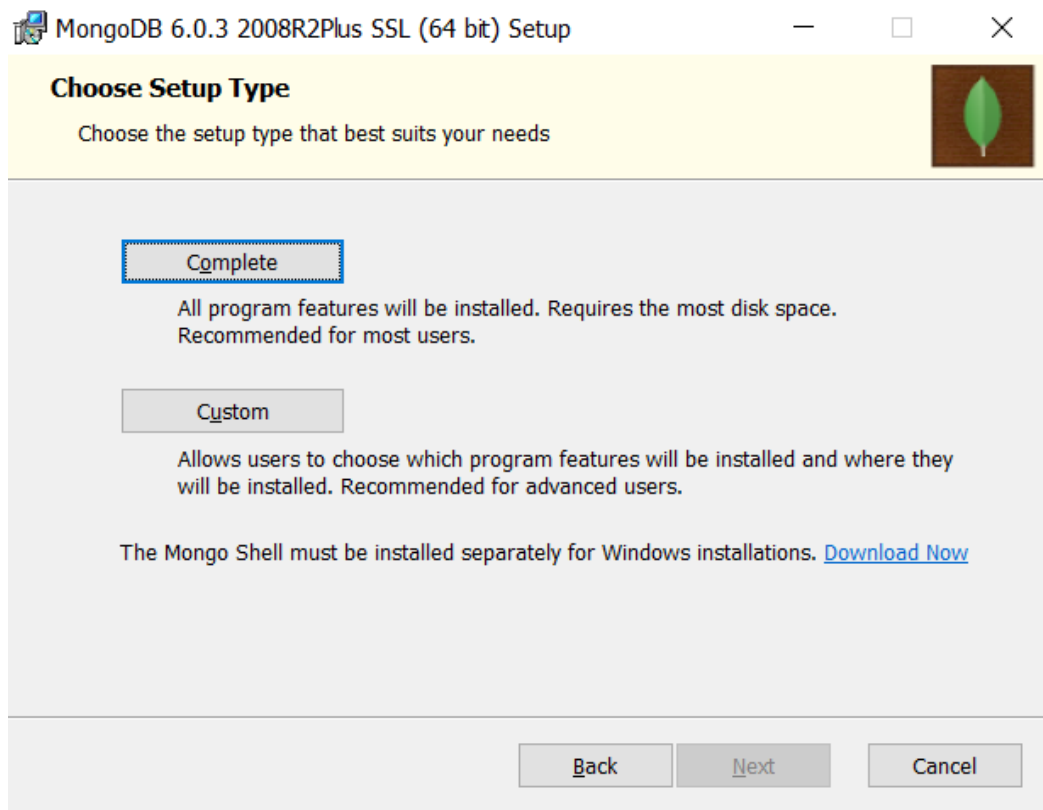


Hình 21. Cài đặt MongoDB (Đồng ý các điều khoản và điều kiện)

- **Bước 4:** Ở đây sẽ có hai option đó là **Complete** hoặc **Custom** (Tùy chọn).
 - Nếu chọn **complete** thì tất cả các chức năng sẽ được cài đặt và tất nhiên là sẽ chiếm nhiều dung lượng ổ cứng hơn.

- Còn nếu chọn **Custom** thì chúng ta sẽ chỉ chọn những chức năng muốn cài đặt và tùy thuộc vào chức năng chúng ta chọn dung lượng ổ cứng cài đặt sẽ giảm đi.

=> Chúng ta thấy MongoDB gợi ý chúng ta (người dùng mới) thì nên cài đặt theo option đầu tiên (Complete). Vì vậy chúng ta cứ chọn Complete và tiếp tục cài đặt.

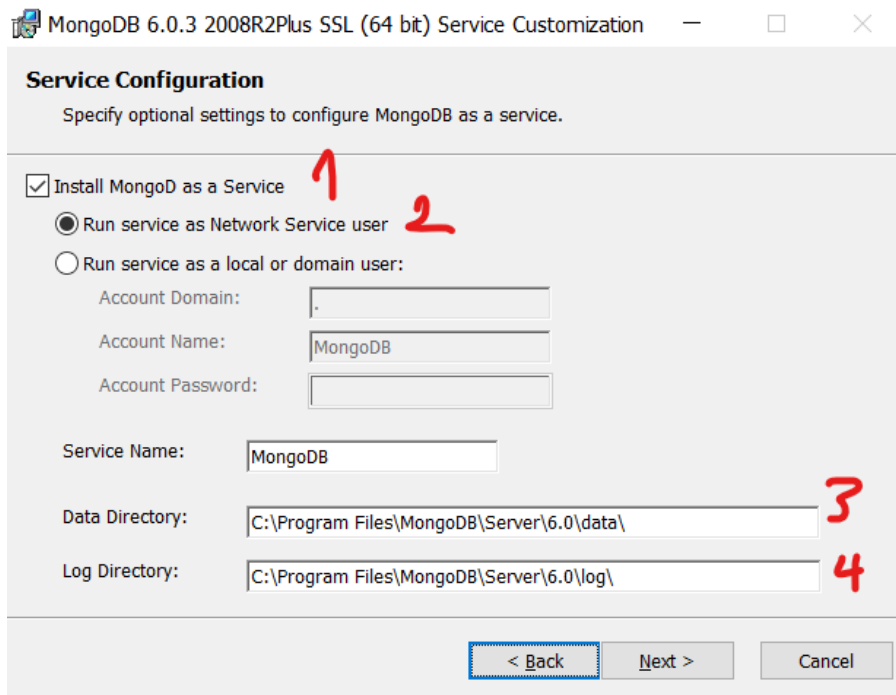


Hình 22. Cài đặt MongoDB (Tùy chọn cài đặt)

- **Bước 5:** Tiếp đến bước cấu hình service.

- (1) Cài đặt MongoDB như là một service. Nó giống như những service bình thường của hệ điều hành. Chúng ta có thể bật/tắt được.
- (2) Chạy service với vai trò là người sử dụng mạng dịch vụ.
- (3) Nơi lưu trữ Database.
- (4) Nơi chứa file log (các server chạy đều có file log, chúng ta có thể xem log file xem server đã chạy các service nào, gặp lỗi ở đâu...)

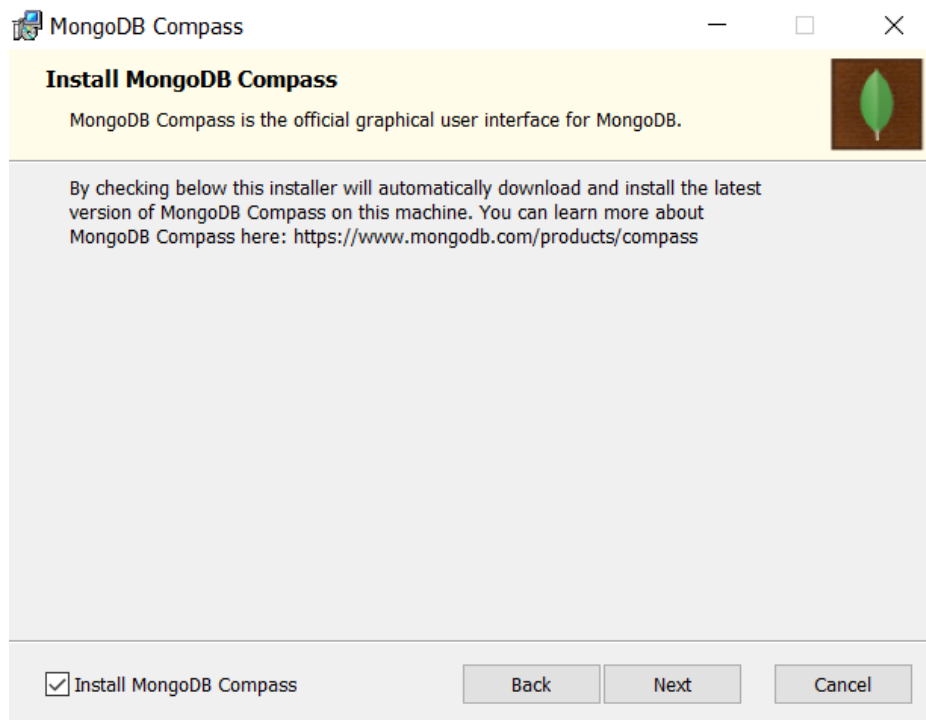
=> Bấm Next để tiếp tục.



Hình 23. Cài đặt MongoDB (Thiết lập cấu hình)

- **Bước 6:** Bước tiếp theo này thì chúng ta sẽ lựa chọn là có cài MongoDB Compass hay không.

MongoDB Compass là một công cụ, chính xác hơn là một giao diện người dùng giúp cho việc thao tác với dữ liệu trong MongoDB trực quan hơn.

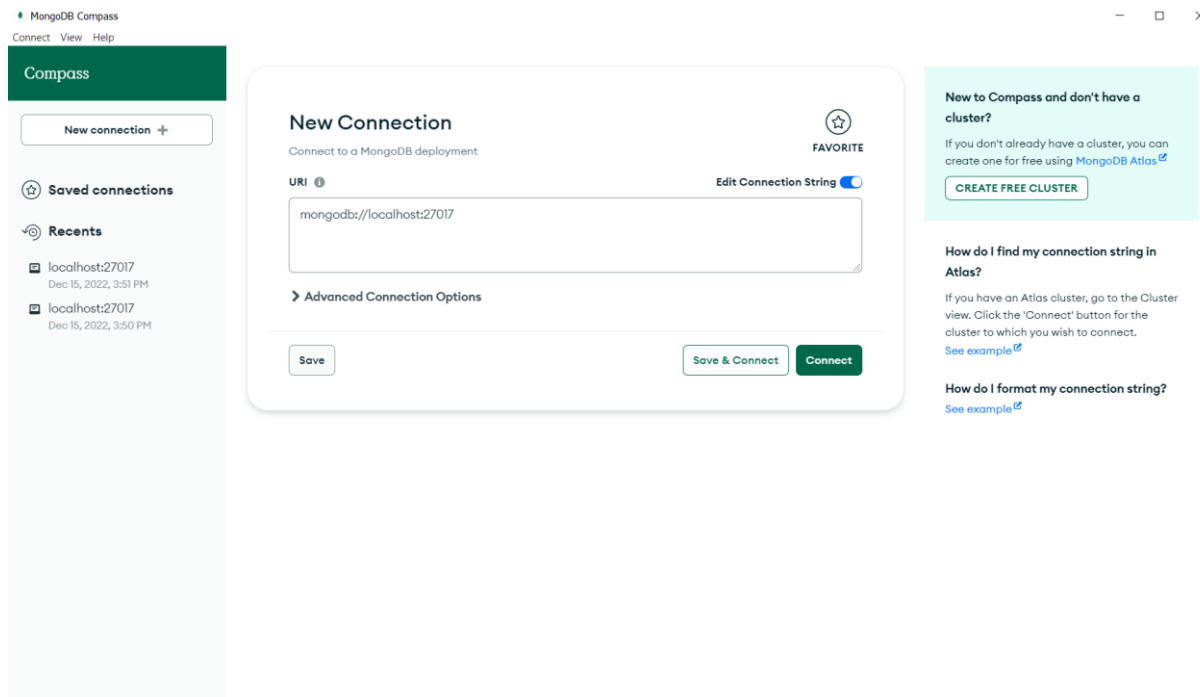


Hình 24. Cài đặt MongoDB (Cài đặt môi trường)

=> Tích và ô **Install MongoDB Compass** để cài đặt => và bấm **Next** để đến bước cài đặt.

- **Bước 7:** Đến bước này thì quá trình cấu hình đã xong và bắt đầu cài đặt. Bấm vào **Install** để bắt đầu quá trình cài đặt. Chờ đến khi cài đặt xong và ấn **Finish**.

Sau khi cài đặt, chạy MongoDB Compass để vào diện như bên dưới. Đây chính là giao diện chúng ta sẽ làm việc với mongodb.



Hình 25. Hoàn tất cài đặt

* *Lưu ý:*

- Truy cập link: <https://www.mongodb.com/docs/manual/administration/install-community/> để xem hướng dẫn chi tiết cách tải và cài đặt trên các hệ điều hành khác như MacOS, Linux.

- Ngoài ra chúng ta có thể cần cài đặt thêm:

+ MongoDB Shell (là một phiên bản đầy đủ chức năng JavaScript và Node.js 16.x `mongosh` REPL môi trường để tương tác với MongoDB deployments. Chúng ta có thể sử dụng MongoDB Shell để kiểm tra các truy vấn và thao tác trực tiếp với cơ sở dữ liệu của bạn.).

Truy cập link: <https://www.mongodb.com/docs/mongodb-shell/install/> để xem hướng dẫn chi tiết hoặc thực hiện theo hình dưới:

Install from MSI

- ➊ Open the MongoDB Shell download page.
Open the [MongoDB Download Center](#).
- ➋ In the **Platform** dropdown, select **Windows 64-bit (8.1+) (MSI)**
- ➌ Click **Download**.
- ➍ Double-click the installer file.
- ➎ Follow the prompts to install `.mongosh`

Hình 26. Các bước cài đặt MongoDB Shell

+ MongoDB Database Tools (là một tập hợp các tiện ích dòng lệnh để làm việc với việc triển khai MongoDB.). Truy cập link: <https://www.mongodb.com/docs/database-tools/installation/installation/> để xem chi tiết hướng dẫn tải và cài đặt.

2. Cài đặt trên cụm máy phân tán

Ở đây chúng tôi sử dụng máy ảo chạy hệ điều hành Ubuntu để hướng dẫn cài đặt cụm máy phân tán.

2.1. Chi tiết cấu hình máy

- Phần mềm máy ảo: VirtualBox
Phiên bản: 6.1.26
- Hệ điều hành: Ubuntu
Phiên bản: 18.04.6 Bionic Beaver
- Phần mềm hỗ trợ kết nối: PuTTY
Phiên bản: 0.78

Thông tin thêm: PuTTY là một triển khai miễn phí SSH và Telnet cho các nền tảng Windows và Unix, và là trình giả lập thiết bị đầu cuối. Để hiểu đây là phần mềm cho phép chúng tôi tạo môi trường giả lập làm việc với Ubuntu trên Window để tiện cho việc thao tác. Mọi thao tác trên PuTTY đều sẽ được lưu và thay đổi trên máy Ubuntu đã được kết nối.

2.2. Các bước thực hiện:

- **Bước 1:** Download các phần mềm

- Truy cập link bên dưới để tải xuống phần mềm máy ảo: VirtualBox

<https://www.virtualbox.org/wiki/Downloads>

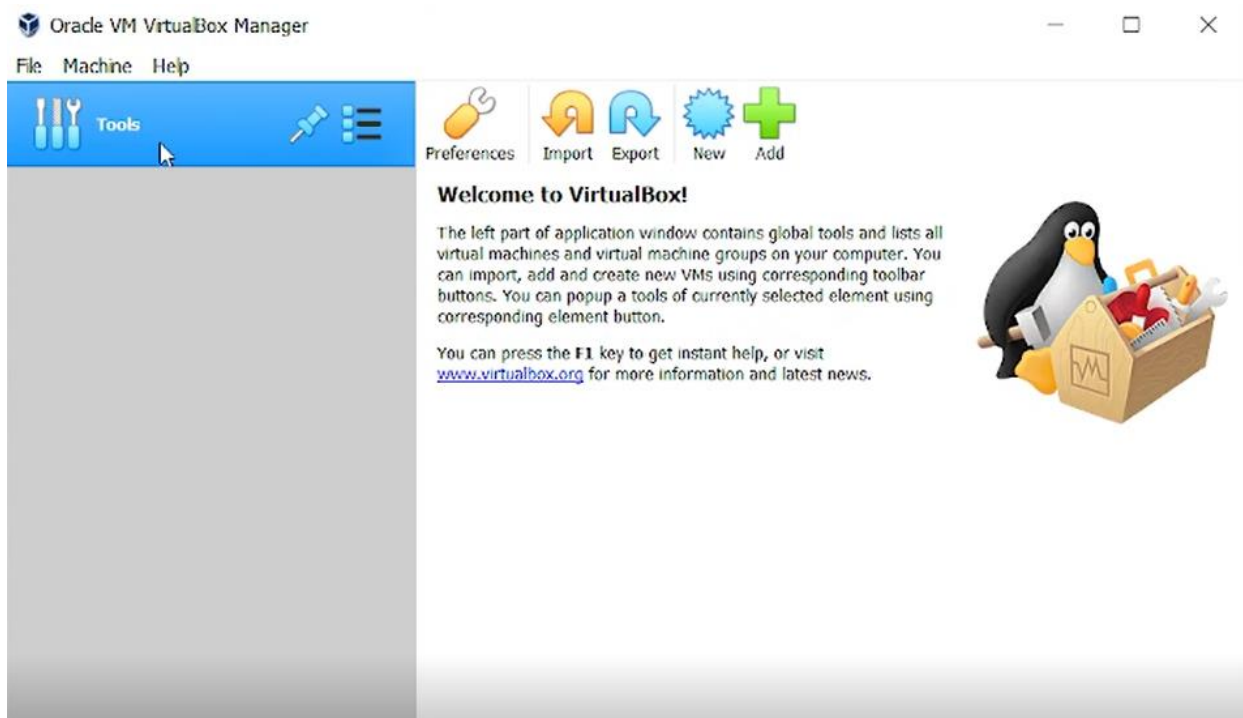
- Truy cập link bên dưới để tải xuống hệ điều hành Ubuntu từ Osboxes

<https://www.osboxes.org/ubuntu/#ubuntu-1804-vbox>

- Truy cập link bên dưới để tải xuống PuTTY

<https://www.putty.org/>

- **Bước 2:** Cài đặt máy ảo VirtualBox

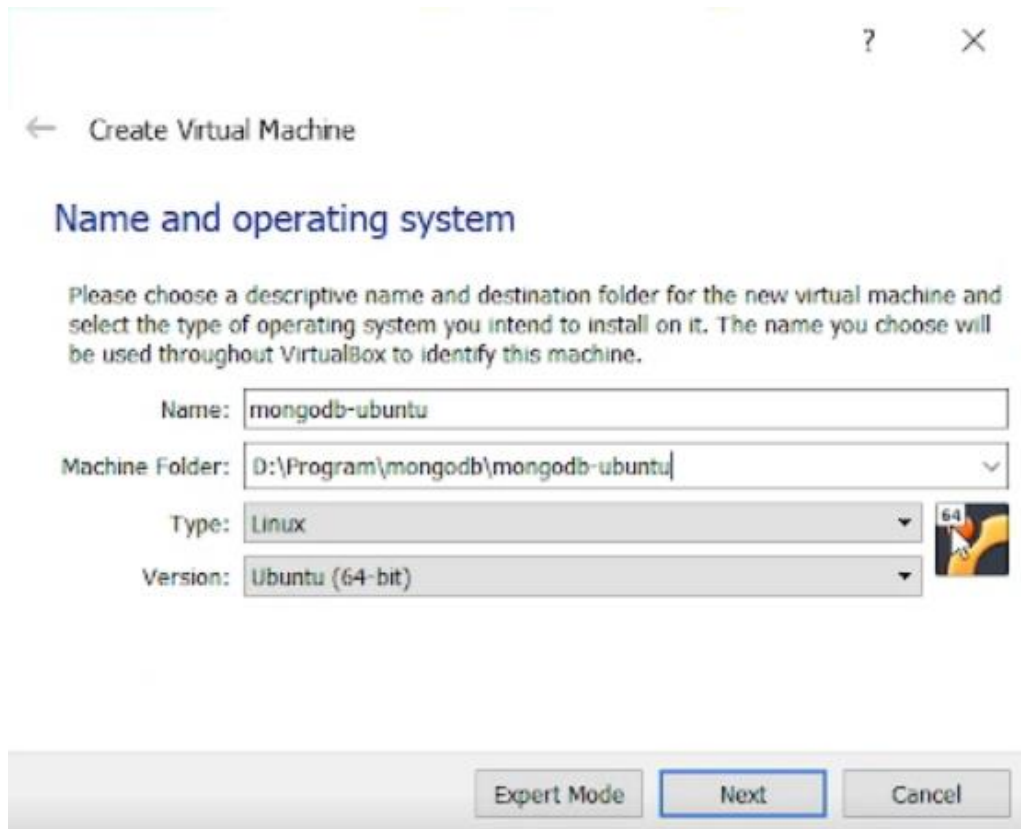


Hình 27. Phần mềm máy ảo VirtualBox

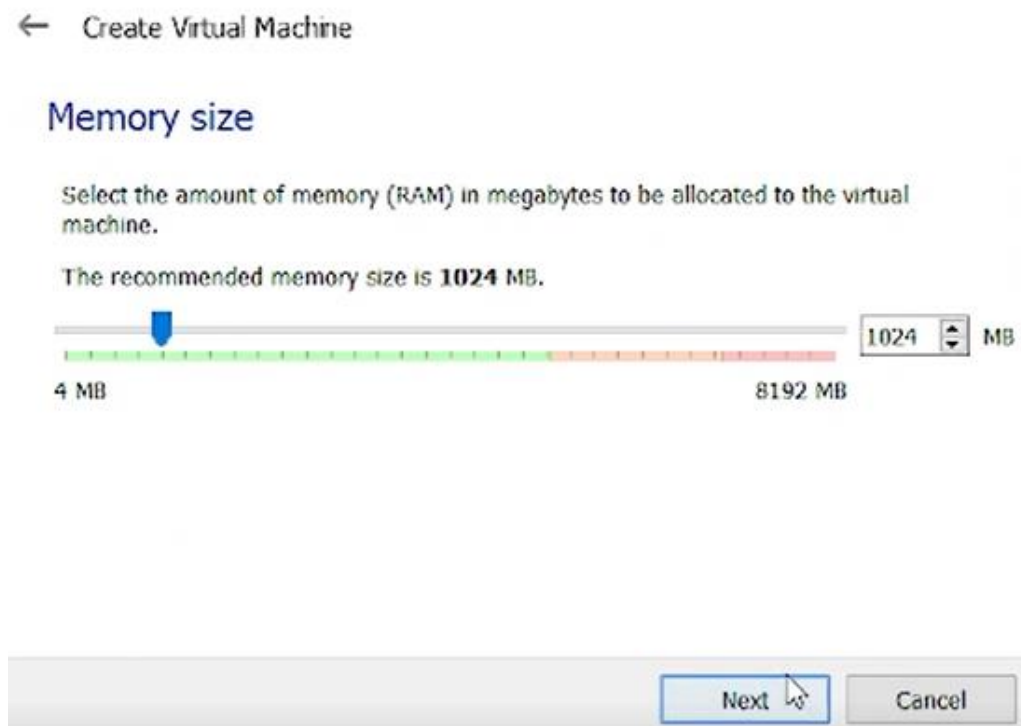
- **Bước 3:** Tạo mới và cấu hình hộp ảo



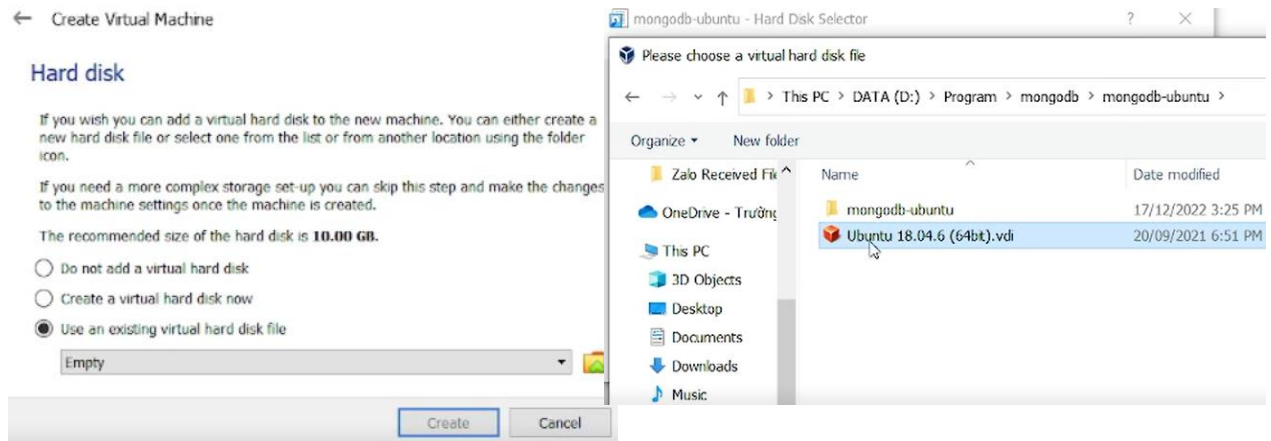
Hình 28. Tạo mới hộp ảo



Hình 29. Thiết lập file nguồn – Type: Linux – Version: Ubuntu (64bit)

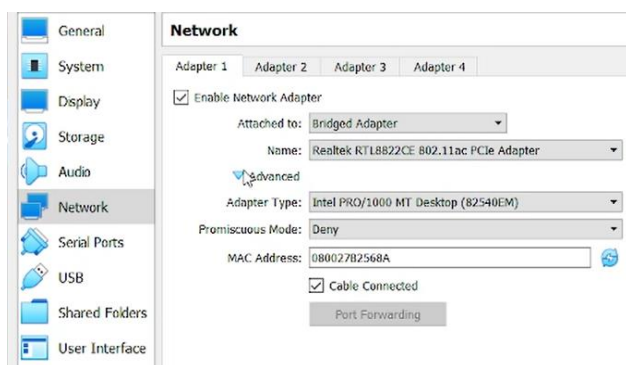


Hình 30. Ram lý tưởng tối thiểu là 1 GB

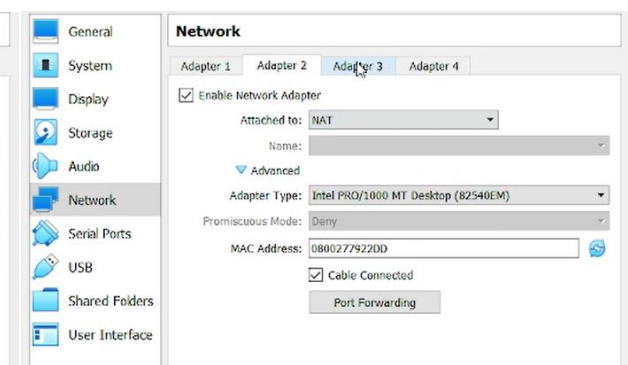


Hình 31. Tùy chọn cài đặt với hệ điều hành Ubuntu đã tải sẵn

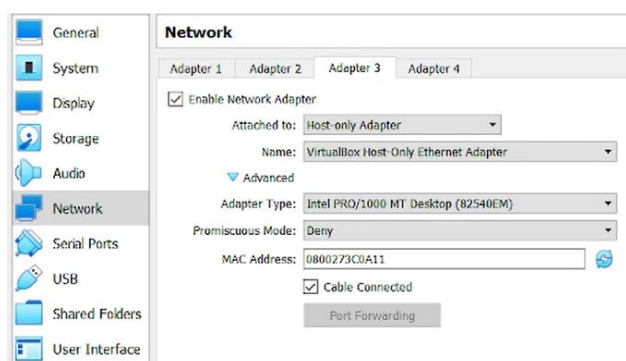
- **Bước 4:** Tùy chỉnh Network và Shared Folders trong phần Setting của hộp ảo
- Adapter 1: Bridged Adapter
- Adapter 2: Nat
- Adapter 3: Host – only Adapter
- Shared Folders: Path là đường dẫn tới thư mục lưu hộp ảo đang tạo, tick vào ô Auto – Mount, Point: /mount/<tên thư mục lưu hộp ảo>



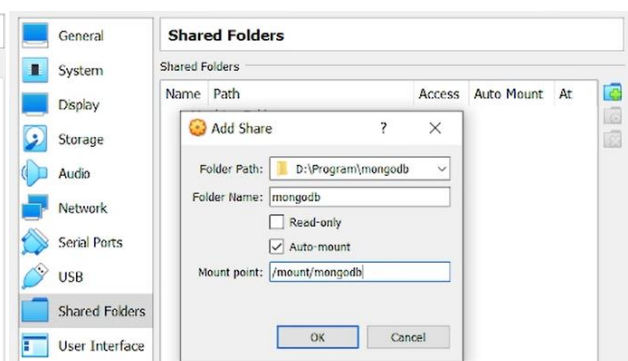
Hình 32. Network – Adapter 1



Hình 33. Network – Adapter 2

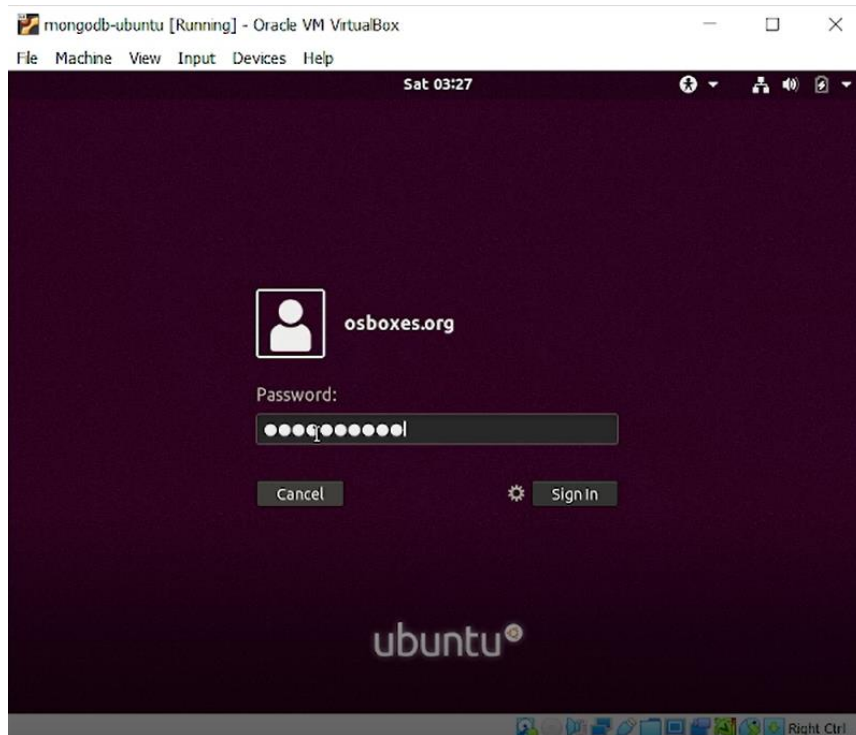


Hình 34. Network – Adapter 3



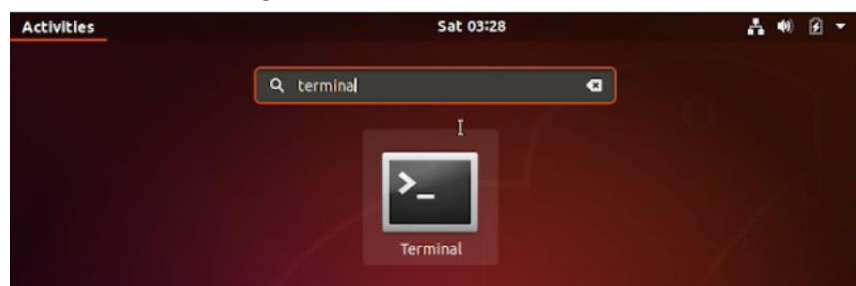
Hình 35. Shared Folders

- **Bước 5:** Khởi chạy máy ảo, đăng nhập với mật khẩu: **osboxes.org**



Hình 36. Khởi động máy ảo và đăng nhập

- **Bước 6:** Khởi động terminal



Hình 37. Khởi động Terminal để thao tác lệnh

- **Bước 7:** Đăng nhập và thực hiện các thao tác lệnh
- Để đăng nhập: Nhập “`sudo su -`” – Mật khẩu: **osboxes.org**
- Kiểm tra Update: có thể thử lệnh “`apt-get update`” để kiểm tra xem có cần cập nhật hay không cùng với kiểm tra truy cập internet trên máy. Ngoài ra, Cài đặt SSH (để kết nối từ PuTTY) bằng cách sử dụng “`apt-get install ssh`”

* *Lưu ý:* Trường hợp gặp lỗi khi chạy lệnh “`apt-get install ssh`”, cách khắc phục như sau: Lần lượt chạy các lệnh sau theo thứ tự để fix:

```
sudo killall apt apt-get
```

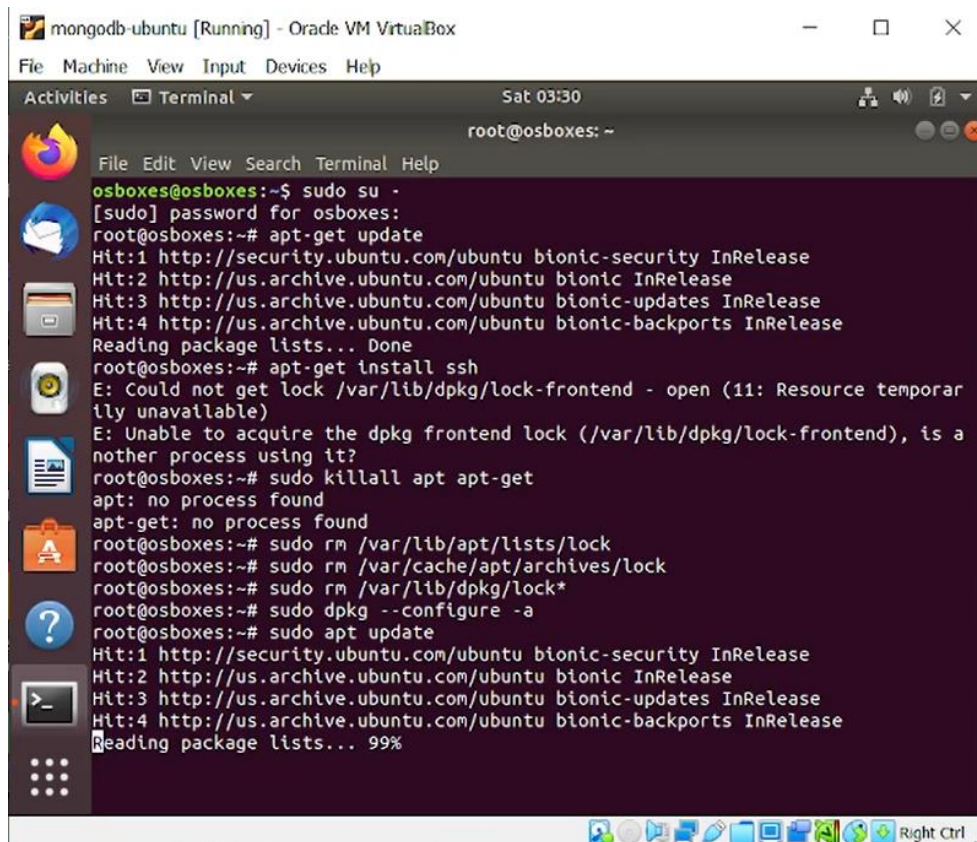
```
sudo rm /var/lib/apt/lists/lock
```

```
sudo rm /var/cache/apt/archives/lock
```

```
sudo rm /var/lib/dpkg/lock*
```

```
sudo dpkg --configure -a
```

```
sudo apt update
```

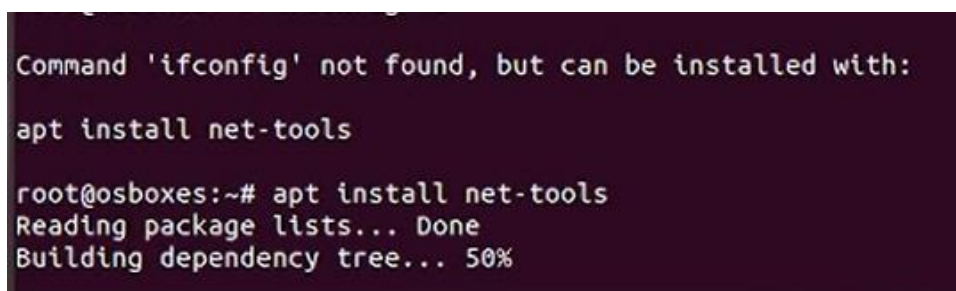


Hình 38. Kiểm tra Update – Cài đặt SSH – Fix lỗi cài đặt

Sau đó thực hiện lại lệnh: “`apt-get install ssh`”

- Kiểm tra IP máy: Sử dụng lệnh “`ifconfig -a`”

* *Lưu ý:* Trường hợp gặp lỗi *not found*, để khắc phục chạy lệnh: “`apt install net-tools`”

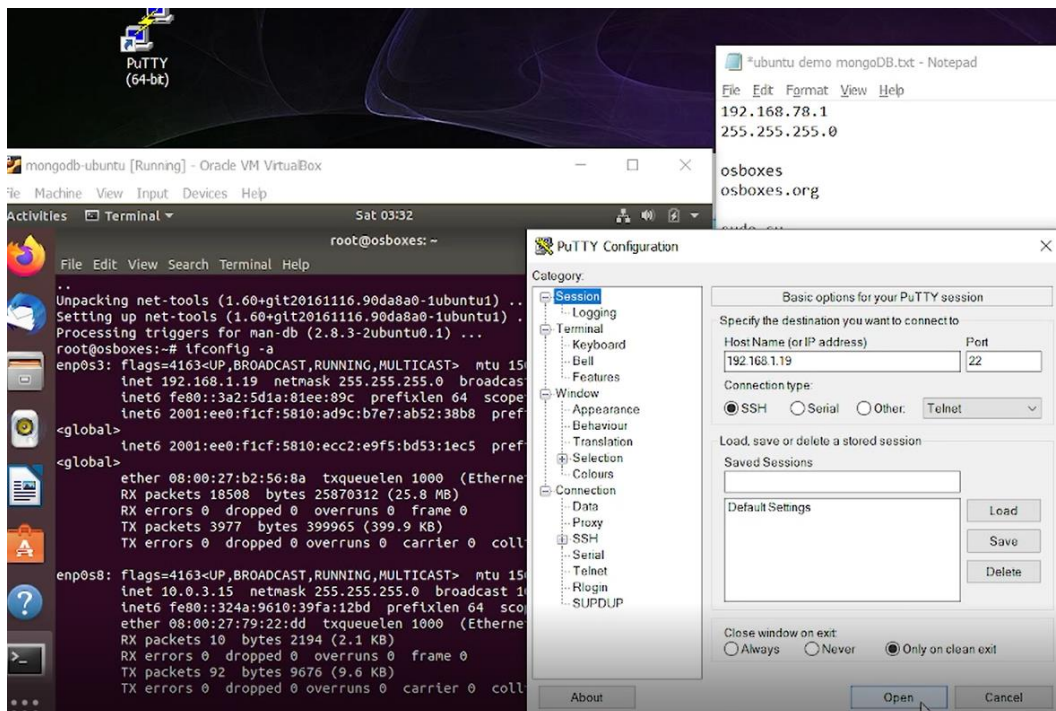


Hình 39. Fix lỗi not found

Sau đó thực hiện lại lệnh: “`ifconfig -a`” để lấy ID

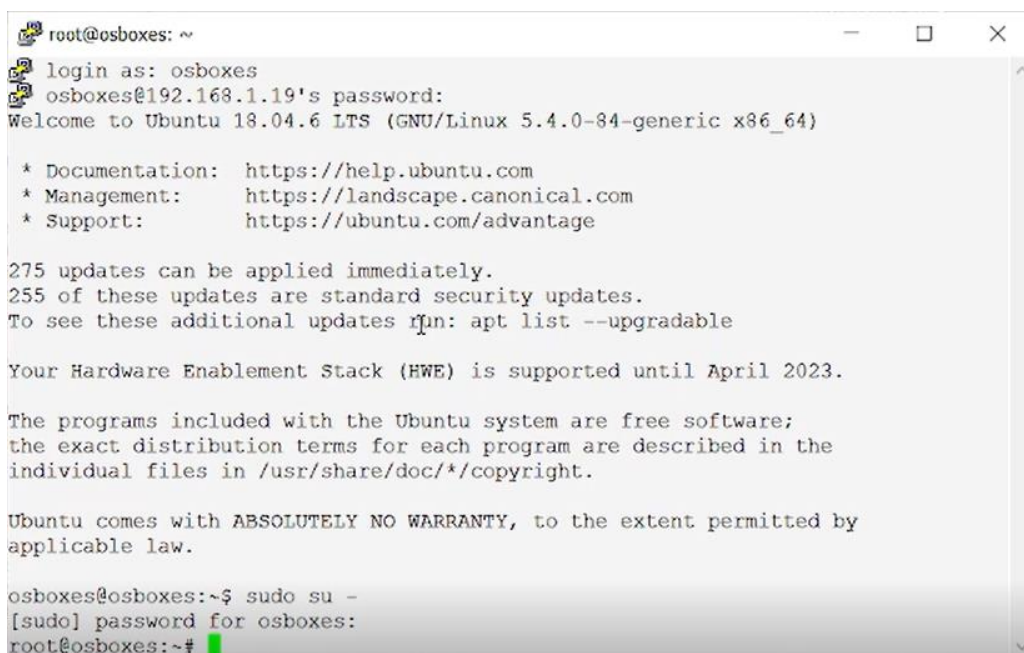
- **Bước 8:** Kết nối máy ảo với PuTTY
- Sau khi kiểm tra ID, chúng ta lấy ID của máy ảo tại đoạn `enp0s3`, dòng 2 sau từ `inet` (như tại hình 21, IP là: (192.168.1.19))

- Mở PuTTY, nhập IP máy ảo Ubuntu vào hộp HostName (or IP address) → Chọn Open và bắt đầu làm việc.



Hình 40. Kiểm tra ID – kết nối PuTTY

- Bước 9:** Cài đặt MongoDB
- Khi hộp thoại PuTTY khởi động sẽ yêu cầu đăng nhập, chúng ta đăng nhập như khi khởi động máy ảo Ubuntu. Tài khoản *login as* mặc định: **osboxes** – mật khẩu: **osboxes.org**
- Thực hiện đăng nhập để thao tác lệnh: **“sudo su - ”** – mật khẩu: **osboxes.org**



Hình 41. Đăng nhập kết nối máy ảo Ubuntu thông qua PuTTY

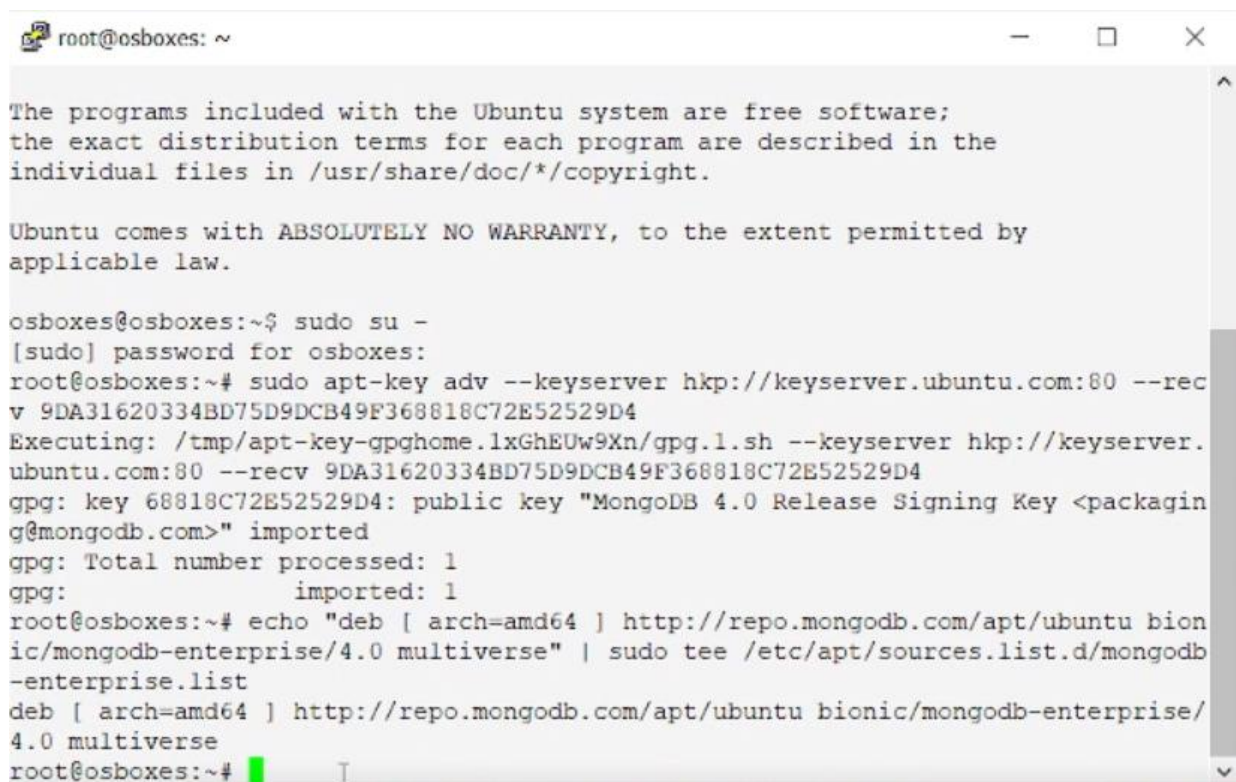
- **Cài đặt MongoDB:** Chúng ta sẽ cài đặt MongoDB cho máy ảo bằng phương thức gói apt-get.. (lệnh)

+ Nhập khóa chung được sử dụng bởi hệ thống quản lý gói:

```
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 9DA31620334BD75D9DCB49F368818C72E52529D4
```

+ Tạo tệp /etc/apt/sources.list.d/mongodb-enterprise.list cho MongoDB

```
echo "deb [ arch=amd64 ] http://repo.mongodb.com/apt/ubuntu bionic/mongodb-enterprise/4.0 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-enterprise.list
```



```
root@osboxes: ~
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

osboxes@osboxes:~$ sudo su -
[sudo] password for osboxes:
root@osboxes:~# sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 9DA31620334BD75D9DCB49F368818C72E52529D4
Executing: /tmp/apt-key-gpghome.lxGhEUw9Xn/gpg.1.sh --keyserver hkp://keyserver.ubuntu.com:80 --recv 9DA31620334BD75D9DCB49F368818C72E52529D4
gpg: key 68818C72E52529D4: public key "MongoDB 4.0 Release Signing Key <packaging@mongodb.com>" imported
gpg: Total number processed: 1
gpg:      imported: 1
root@osboxes:~# echo "deb [ arch=amd64 ] http://repo.mongodb.com/apt/ubuntu bionic/mongodb-enterprise/4.0 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-enterprise.list
deb [ arch=amd64 ] http://repo.mongodb.com/apt/ubuntu bionic/mongodb-enterprise/4.0 multiverse
root@osboxes:~#
```

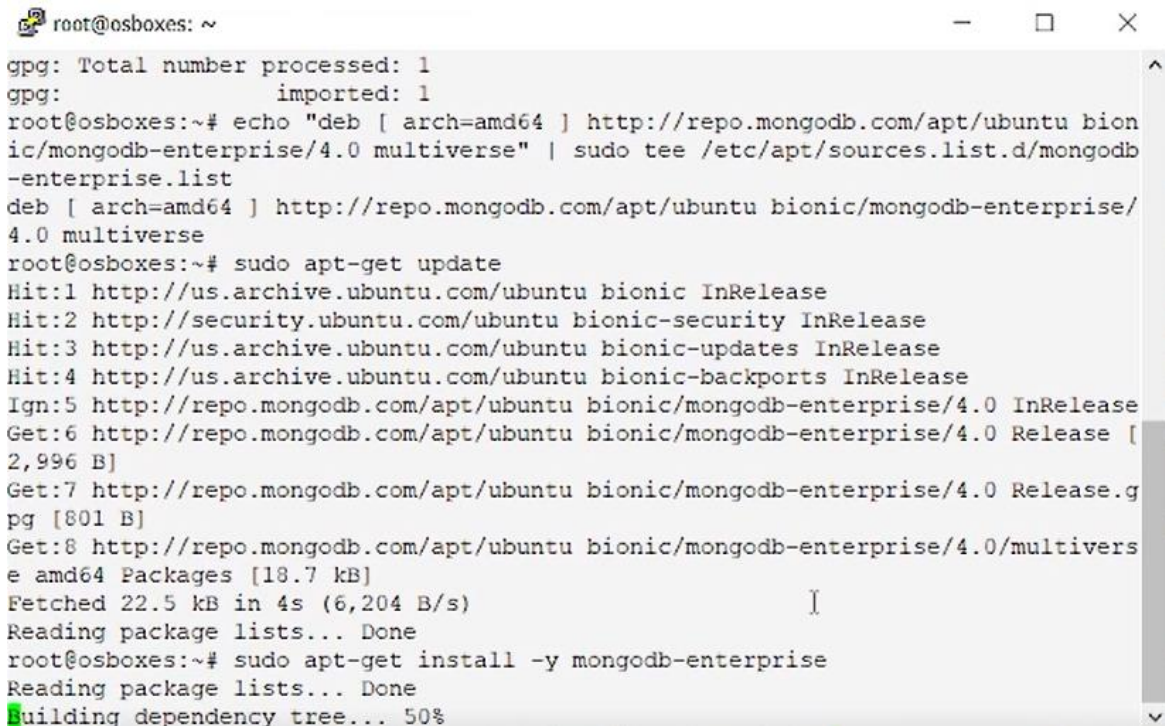
Hình 42. Nhập khóa và thiết lập để cài đặt MongoDB cho máy ảo

+ Reload local package database:

```
sudo apt-get update
```

+ Cài đặt gói MongoDB:

```
sudo apt-get install -y mongodb-enterprise
```

```

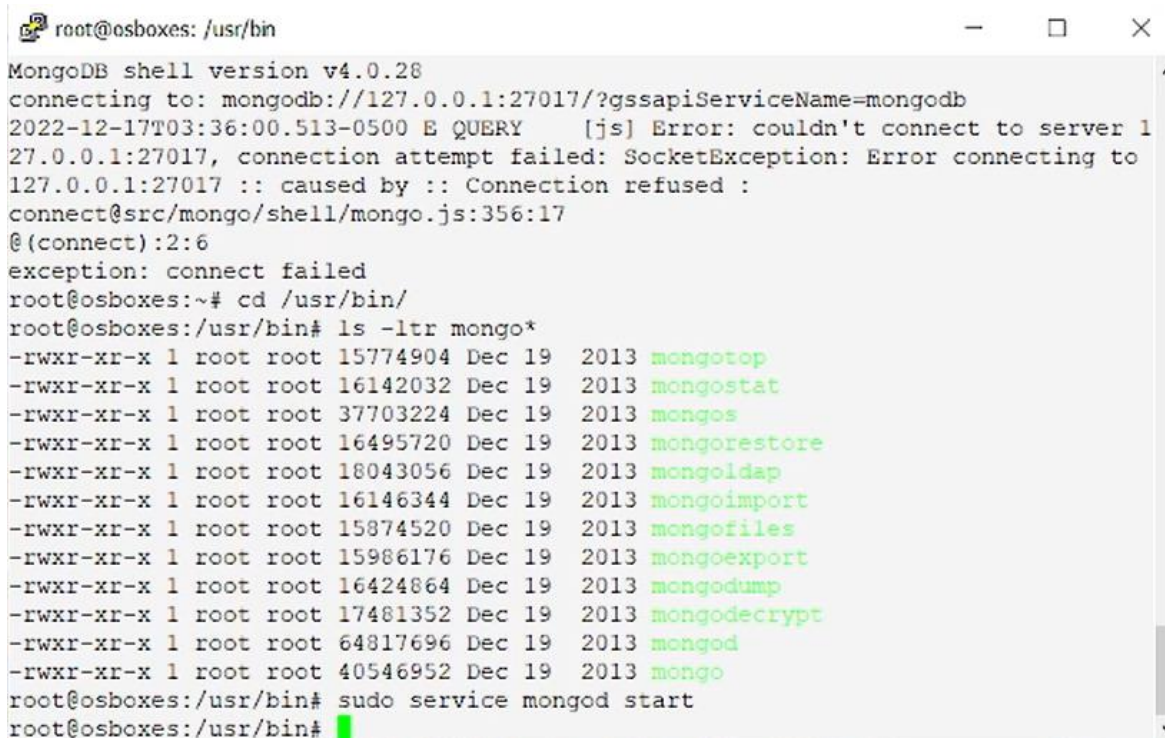
root@osboxes: ~
gpg: Total number processed: 1
gpg: imported: 1
root@osboxes:~# echo "deb [ arch=amd64 ] http://repo.mongodb.com/apt/ubuntu bionic/mongodb-enterprise/4.0 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-enterprise.list
deb [ arch=amd64 ] http://repo.mongodb.com/apt/ubuntu bionic/mongodb-enterprise/4.0 multiverse
root@osboxes:~# sudo apt-get update
Hit:1 http://us.archive.ubuntu.com/ubuntu bionic InRelease
Hit:2 http://security.ubuntu.com/ubuntu bionic-security InRelease
Hit:3 http://us.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:4 http://us.archive.ubuntu.com/ubuntu bionic-backports InRelease
Ign:5 http://repo.mongodb.com/apt/ubuntu bionic/mongodb-enterprise/4.0 InRelease
Get:6 http://repo.mongodb.com/apt/ubuntu bionic/mongodb-enterprise/4.0 Release [2,996 B]
Get:7 http://repo.mongodb.com/apt/ubuntu bionic/mongodb-enterprise/4.0 Release.gpg [801 B]
Get:8 http://repo.mongodb.com/apt/ubuntu bionic/mongodb-enterprise/4.0/multiverse amd64 Packages [18.7 kB]
Fetched 22.5 kB in 4s (6,204 B/s)
Reading package lists... Done
root@osboxes:~# sudo apt-get install -y mongodb-enterprise
Reading package lists... Done
Building dependency tree... 50%

```

Hình 43. Cài đặt gói MongoDB

+ Chúng ta cũng có thể xem danh sách các nhị phân mongo được cài đặt và kiểm tra xem mongo đã cài đặt đúng chưa bằng lệnh sau:

```
cd /usr/bin
ls -ltr mongo*
```



```

root@osboxes: /usr/bin
MongoDB shell version v4.0.28
connecting to: mongodb://127.0.0.1:27017/?gssapiServiceName=mongodb
2022-12-17T03:36:00.513-0500 E QUERY [js] Error: couldn't connect to server 127.0.0.1:27017, connection attempt failed: SocketException: Error connecting to 127.0.0.1:27017 :: caused by :: Connection refused :
connect@src/mongo/shell/mongo.js:356:17
@(connect):2:6
exception: connect failed
root@osboxes:~# cd /usr/bin/
root@osboxes:/usr/bin# ls -ltr mongo*
-rwxr-xr-x 1 root root 15774904 Dec 19 2013 mongotop
-rwxr-xr-x 1 root root 16142032 Dec 19 2013 mongostat
-rwxr-xr-x 1 root root 37703224 Dec 19 2013 mongos
-rwxr-xr-x 1 root root 16495720 Dec 19 2013 mongorestore
-rwxr-xr-x 1 root root 18043056 Dec 19 2013 mongodap
-rwxr-xr-x 1 root root 16146344 Dec 19 2013 mongoimport
-rwxr-xr-x 1 root root 15874520 Dec 19 2013 mongofiles
-rwxr-xr-x 1 root root 15986176 Dec 19 2013 mongoexport
-rwxr-xr-x 1 root root 16424864 Dec 19 2013 mongodump
-rwxr-xr-x 1 root root 17481352 Dec 19 2013 mongodecrypt
-rwxr-xr-x 1 root root 64817696 Dec 19 2013 mongod
-rwxr-xr-x 1 root root 40546952 Dec 19 2013 mongo
root@osboxes:/usr/bin# sudo service mongod start
root@osboxes:/usr/bin#

```

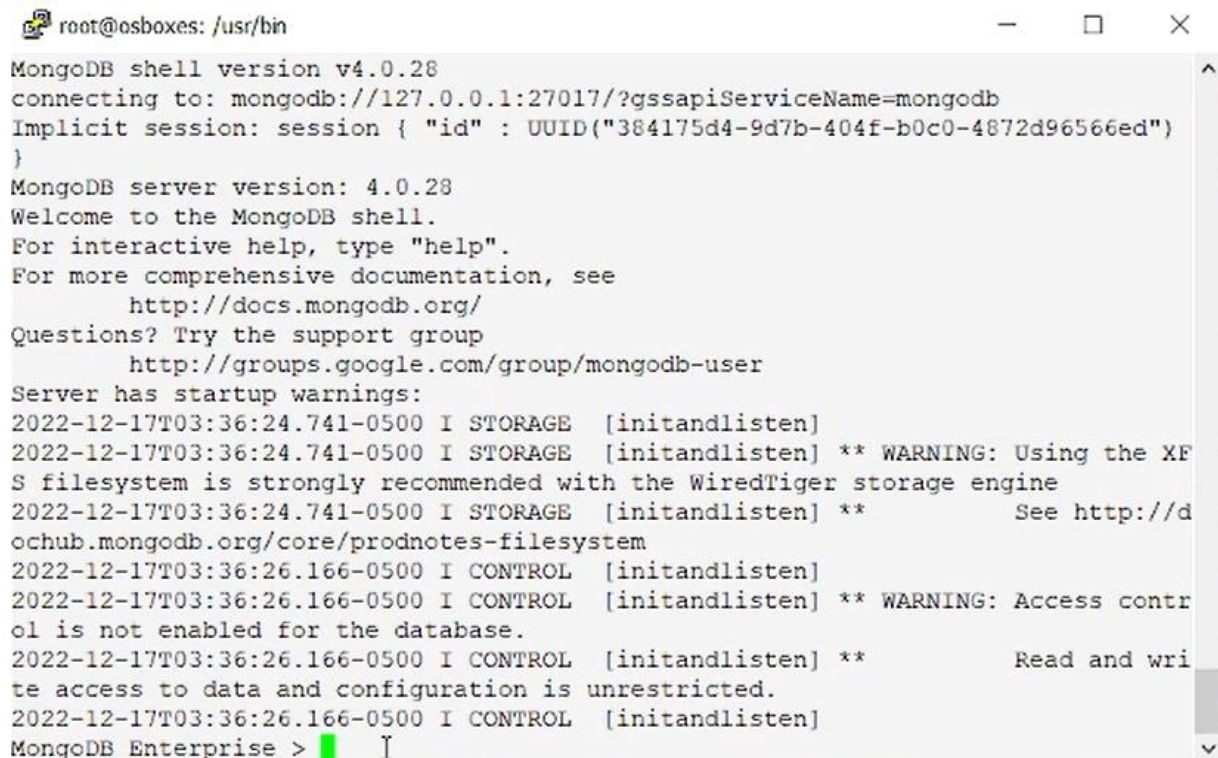
Hình 44. Kiểm tra sau khi cài đặt

- **Bước 10:** Khởi động MongoDB

+ Khởi động service MongoDB, chỉ cần gõ:

```
sudo service mongod start
```

+ Kết nối với mongo shell bằng lệnh: `mongo`



```

root@osboxes: /usr/bin
MongoDB shell version v4.0.28
connecting to: mongodb://127.0.0.1:27017/?gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("384175d4-9d7b-404f-b0c0-4872d96566ed")
}
MongoDB server version: 4.0.28
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
  http://docs.mongodb.org/
Questions? Try the support group
  http://groups.google.com/group/mongodb-user
Server has startup warnings:
2022-12-17T03:36:24.741-0500 I STORAGE  [initandlisten]
2022-12-17T03:36:24.741-0500 I STORAGE  [initandlisten] ** WARNING: Using the XFS
filesystem is strongly recommended with the WiredTiger storage engine
2022-12-17T03:36:24.741-0500 I STORAGE  [initandlisten] **           See http://d
ochub.mongodb.org/core/prodnotes-filesystem
2022-12-17T03:36:26.166-0500 I CONTROL  [initandlisten]
2022-12-17T03:36:26.166-0500 I CONTROL  [initandlisten] ** WARNING: Access contr
ol is not enabled for the database.
2022-12-17T03:36:26.166-0500 I CONTROL  [initandlisten] **           Read and wri
te access to data and configuration is unrestricted.
2022-12-17T03:36:26.166-0500 I CONTROL  [initandlisten]
MongoDB Enterprise >

```

Hình 45. Khởi động MongoDB Shell

- **Bước 11:** Cấu hình tập tin `mongod.conf`

+ Để mở tập tin `mongod.conf`, nhập lệnh:

```
vi /etc/mongod.conf
```

+ Thay thế: Port 127.0.0.1 → thành 0.0.0.0

+ Bỏ dấu # trước dòng `repllocation`:

+ Thêm dòng: **`replSetName: "g13demo"`**

- Sau đó khởi động lại service MongoDB:

```
sudo service mongod start
```

```

dbPath: /var/lib/mongodb
journal:
  enabled: true
# engine:
# mmapv1:
# wiredTiger:

# where to write logging data.
systemLog:
  destination: file
  logAppend: true
  path: /var/log/mongodb/mongod.log

# network interfaces
net:
  port: 27017
  bindIp: 0.0.0.0

# how the process runs
processManagement:
  timeZoneInfo: /usr/share/zoneinfo

#security:

#operationProfiling:

replication:
  replSetName: "g13demo"

#sharding:

:wq

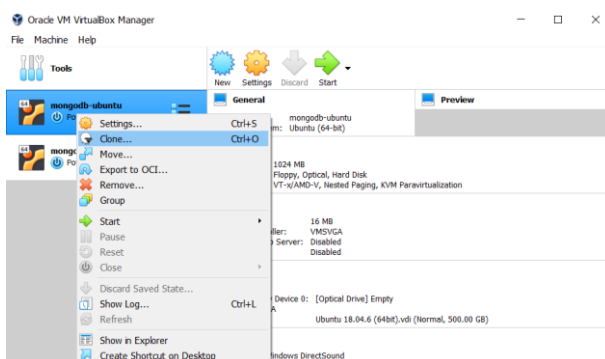
```

Hình 46. Cấu hình tập tin mongod.conf

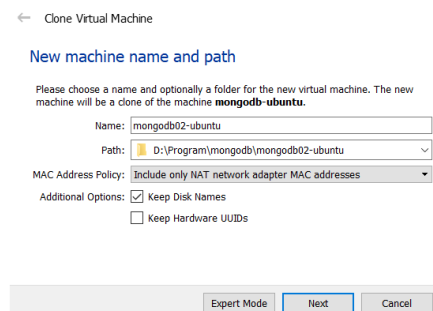
- Bước 12: Cấu hình các máy trong cụm phân tán

* Tạo thêm máy ảo

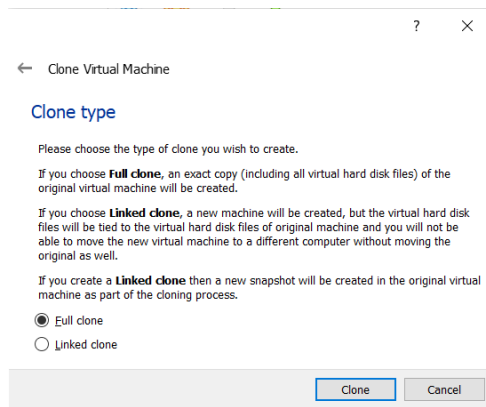
- Cách 1: Thực hiện như từ Bước 1 → Bước 9
- Cách 2: Sử dụng chức năng tạo clone của máy ảo VirtualBox. Chúng tôi đề xuất thực hiện theo cách này để tiết kiệm thời gian. Chi tiết thực hiện như sau:
 - + Bước 1: Tắt và lưu lại thiết lập máy ảo vừa thiết lập
 - + Bước 2: Chuột phải vào máy ảo → chọn Clone
 - + Bước 3: Thay name phù hợp, Path: Tạo folder mới lưu trong file cài đặt máy ảo đầu tiên, đổi tên phù hợp và trở đến. Tích vào ô Keep Disk Names → chọn Next
 - + Bước 4: Tích vào ô Full Clone → chọn Clone



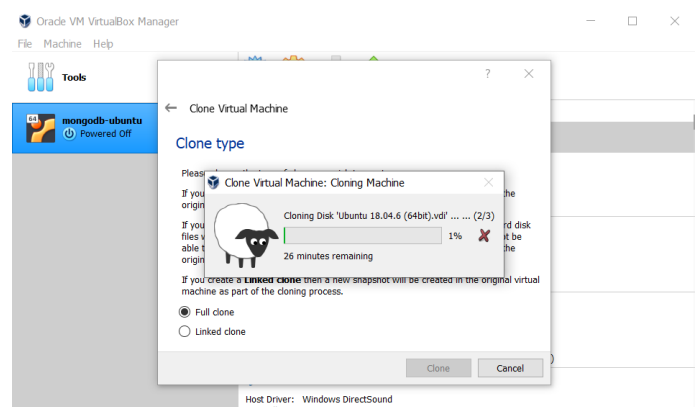
Hình 47. Chọn Clone máy ảo đã thiết lập



Hình 48. Thay đổi tập tin lưu trữ



Hình 49. Chọn Full clone



Hình 50. Bắt đầu tiến trình clone

* Lưu ý:

Tài khoản và mật khẩu của các máy ảo sau khi tạo đều như nhau.

- **Bước 13:** Đổi Hostname các máy trong cụm phân tán và chỉ định IP các máy trong tập tin Hosts.

+ Để đổi Hostname, trước tiên đăng nhập vào bằng lệnh: `sudo su -`, nhập lệnh sau:

`hostnamectl set-hostname <tên>`, sau đó nhập `hostnamectl` để kiểm tra.

```
osboxes@tphcm: ~
177 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

New release '20.04.5 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Your Hardware Enablement Stack (HWE) is supported until April 2023.
Last login: Sat Dec 17 03:47:04 2022 from 192.168.1.44
osboxes@tphcm:~$ sudo su -
[sudo] password for osboxes:
root@tphcm:~# hostnamectl set-hostname hanoi
root@tphcm:~# hostnamectl
  Static hostname: hanoi
            Icon name: computer-vm
            Chassis: vm
            Machine ID: 59bad714bed846048537ec1ce8c3a4d9
            Boot ID: 5956f3f1bec74aebb051bab8a1001267
    Virtualization: oracle
  Operating System: Ubuntu 18.04.6 LTS
            Kernel: Linux 5.4.0-84-generic
    Architecture: x86_64
```

Hình 51. Thay đổi hostname

+ Để chỉ định IP các máy trong tập tin Hosts, để mở tập tin Hosts nhập lệnh:

`vi /etc/host`

+ Thêm địa chỉ IP vào theo cú pháp sau:

[Địa chỉ IP] <hostname máy 1>.localdomain <hostname máy 1>

[Địa chỉ IP] <hostname máy 2>.localdomain <hostname máy 2>

...

+ Ví dụ:

192.168.1.19 tphcm.localdomain tphcm

192.168.1.22 hanoi.localdomain hanoi

```
192.168.1.19    tphcm.localdomain    tphcm
192.168.1.22    hanoi.localdomain    hanoi

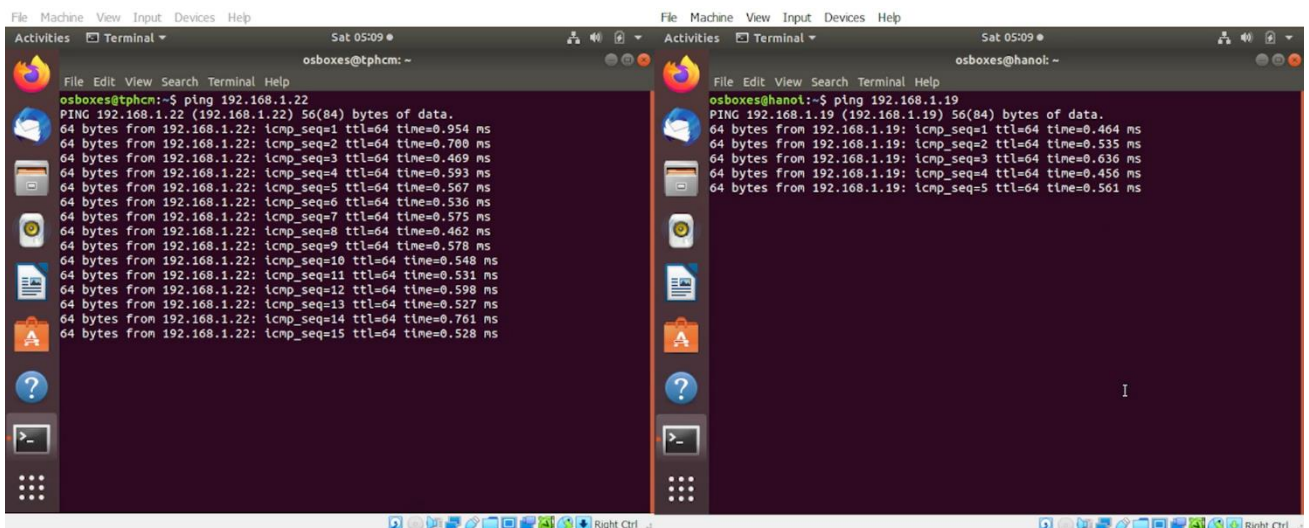
127.0.0.1      localhost
127.0.1.1      osboxes

# The following lines are desirable for IPv6 capable hosts
::1            ip6-localhost ip6-loopback
fe00::0        ip6-localnet
ff00::0        ip6-mcastprefix
ff02::1        ip6-allnodes
ff02::2        ip6-allrouters
~
~
```

Hình 52. Thêm chỉ định IP trong tập tin Hosts

- **Bước 14:** Kiểm tra kết nối các máy.

Kiểm tra xem các máy đang có thể kết nối với nhau không bằng lệnh: `ping <IP máy khác cần kết nối>`



Hình 53. Ping IP để kiểm tra kết nối

- **Bước 15:** Thiết lập liên kết phân tán Database.

Ở đây chúng tôi sử dụng cơ chế Replica Set Member của MongoDB để thiết lập cụm phân tán. Cơ chế này khá giống với cơ chế Master – Slave, tuy nhiên nó khác ở chỗ role

Master là Primary và role Slave là các Secondary. Ngoài ra cơ chế Replica Set Member còn có thêm role Arbiter (hay còn gọi là trọng tài).

Cơ chế này chỉ cho phép role Primary có thể thực hiện toàn bộ các thao tác lên database lưu trữ trong cụm phân tán, các Secondary chỉ có chức năng truy vấn (read), còn các Arbiter chỉ mang chức năng chính là làm trọng tài và không lưu trữ database cũng như không có bất cứ quyền nào lên Database. Điều khác biệt là cơ chế này khi Primary xảy ra sự cố hoặc chủ động từ chức, một cuộc bầu cử sẽ diễn ra để chọn 1 trong các Secondary lên thay, và khi đó trọng tài Arbiter sẽ một trong các yếu tố quyết định Secondary nào sẽ lên làm Primary, khi đó Primary cũ sẽ bị hạ chức thành Secondary.

Chúng tôi sẽ hướng dẫn tạo liên kết cụm phân tán và cài đặt Arbiter.

+ Tạo liên kết cụm phân tán:

- Bước 1: Khởi động MongoDB Shell ở cả 2 máy
- Bước 2: Ở máy chủ được chỉ định làm Primary, chúng ta nhập lệnh sau:

```
rs.initiate( {
  _id : "g13demo",
  members: [
    { _id: 0, host: "192.168.1.19:27017" },
    { _id: 1, host: "192.168.1.22:27017" },
  ]
})
```

* *Giải thích:*

- **"g13demo":** tên liên kết, được thiết lập ở dòng replSetName ở tập mongod.conf
- **members:** Danh sách các thành viên sẽ đưa vào cụm phân tán, cú pháp các thành viên như sau: { **_id**: <số id>, **host**: <địa chỉ IP và port của máy thành viên> }
- Bước 3: Chờ một lát để máy các máy kết nối với nhau và tạo liên kết phân tán
- Bước 4: Ấn Enter ở các máy để xem kết quả thể hiện tên liên kết và role.


```

root@tphcm:~# mongo
MongoDB shell version v4.0.28
connecting to: mongodb://127.0.0.1:27017/?qssapiServiceName=mongodb
Implicit session: session { "id" : UUID("ae070e21-d713-4acd-b573-42ef5905bc8b") }
MongoDB server version: 4.0.28
Server has startup warnings:
2022-12-17T05:10:58.941-0500 I STORAGE [initandlisten]
2022-12-17T05:10:58.941-0500 I STORAGE [initandlisten] ** WARNING: Using the XFS file
system is strongly recommended with the WiredTiger storage engine
2022-12-17T05:10:58.941-0500 I STORAGE [initandlisten] ** See http://dochub.
mongodb.org/core/prodnotes-filesystem
2022-12-17T05:10:59.980-0500 I CONTROL [initandlisten]
2022-12-17T05:10:59.980-0500 I CONTROL [initandlisten] ** WARNING: Access control is
not enabled for the database.
2022-12-17T05:10:59.980-0500 I CONTROL [initandlisten] ** Read and write acc
ess to data and configuration is unrestricted.
2022-12-17T05:10:59.980-0500 I CONTROL [initandlisten]
MongoDB Enterprise > rs.initiate( {
...   _id : "gl3demo",
...   members: [
...     { _id: 0, host: "192.168.1.19:27017" },
...     { _id: 1, host: "192.168.1.22:27017" },
...   ]
... } )
{ "ok" : 1 }
MongoDB Enterprise gl3demo:OTHER>
MongoDB Enterprise gl3demo:SECONDARY>
MongoDB Enterprise gl3demo:SECONDARY>
MongoDB Enterprise gl3demo:PRIMARY>

root@hanoi:~# mongo
MongoDB shell version v4.0.28
connecting to: mongodb://127.0.0.1:27017/?qssapiServiceName=mongodb
Implicit session: session { "id" : UUID("888c3511-a55e-4724-b7a3-fccbbab4a989") }
MongoDB server version: 4.0.28
Server has startup warnings:
2022-12-17T05:11:06.761-0500 I STORAGE [initandlisten]
2022-12-17T05:11:06.761-0500 I STORAGE [initandlisten] ** WARNING: Using the XFS file
system is strongly recommended with the WiredTiger storage engine
2022-12-17T05:11:06.761-0500 I STORAGE [initandlisten] ** See http://dochub.
mongodb.org/core/prodnotes-filesystem
2022-12-17T05:11:07.866-0500 I CONTROL [initandlisten]
2022-12-17T05:11:07.866-0500 I CONTROL [initandlisten] ** WARNING: Access control is
not enabled for the database.
2022-12-17T05:11:07.866-0500 I CONTROL [initandlisten] ** Read and write acc
ess to data and configuration is unrestricted.
2022-12-17T05:11:07.866-0500 I CONTROL [initandlisten]
MongoDB Enterprise >
MongoDB Enterprise >
MongoDB Enterprise >
MongoDB Enterprise gl3demo:SECONDARY>
MongoDB Enterprise gl3demo:SECONDARY>
MongoDB Enterprise gl3demo:SECONDARY>

```

Hình 54. Kết nối phân tán giữa các máy

+ Cài đặt Arbiter:

- Bước 1: Mở thêm một MongoDB Shell của một máy Secondary
- Bước 2: sao chép tập tin `/etc/mongod.conf` sang `/etc/mongod_arb.conf` và thực hiện các thay đổi trong tập mới.

```

root@hanoi:~# cp /etc/mongod.conf /etc/mongod_arb.conf
root@hanoi:~# vi /etc/mongod_arb.conf

[1]+  Stopped                  vi /etc/mongod_arb.conf
root@hanoi:~# vi /etc/mongod_arb.conf

```

Hình 55. Sao chép tập `mongod.conf` thành `mongod_arb.conf`

- Cập nhật `db_path` thành `“/var/lib/mongodb_arb”`
- Cập nhật `Log_path` thành `“/var/log/mongodb/mongod_arb.log”`
- Cập nhật `Port` thành bất kỳ (vd: 27018)

```

storage:
  dbPath: /var/lib/mongodb_arb
  journal:
    enabled: true
  # engine:
  # mmapv1:
  # wiredTiger:

# where to write logging data.
systemLog:
  destination: file
  logAppend: true
  path: /var/log/mongodb/mongod_arb.log

# network interfaces
net:
  port: 27017

```

Hình 56. Cập nhật trong tập `mongod_arb.conf`

- Bước 3: Tạo và thiết lập tệp “/var/lib/mongodb_Arb” để lưu trữ các tệp mongo bằng lệnh:

```
mkdir /var/lib/mongodb_arb
```

```
root@hanoi:~# cp /etc/mongod.conf /etc/mongod_arb.conf
root@hanoi:~# vi /etc/mongod_arb.conf

[1]+  Stopped                  vi /etc/mongod_arb.conf
root@hanoi:~# vi /etc/mongod_arb.conf
root@hanoi:~# mkdir /var/lib/mongodb_arb
root@hanoi:~#
root@hanoi:~# mongod --config /etc/mongod_arb.conf
```

Hình 57. Tạo và thiết lập “/var/lib/mongodb_Arb”

- Bước 4: Khi các thay đổi đã hoàn tất, hãy bắt đầu phiên bản Arbiter bằng cách sử dụng lệnh “mongod --config /etc/mongod_arb.conf” và “mongod --config /etc/mongod_arb.conf &”. Khi Arbiter khởi chạy, hãy kết nối bằng cách sử dụng “mongo --port 27018”

```
root@hanoi:~# mongod --config /etc/mongod_arb.conf
root@hanoi:~# mongod --config /etc/mongod_arb.conf &
[2] 28858
root@hanoi:~# mongo --port 27018
MongoDB shell version v4.0.28
connecting to: mongodb://127.0.0.1:27018/?gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("c88f3d13-b0ae-41c6-b4b4-15b5eefff8c2")
}
MongoDB server version: 4.0.28
Server has startup warnings:
2022-12-17T09:11:38.101-0500 I STORAGE [initandlisten]
2022-12-17T09:11:38.101-0500 I STORAGE [initandlisten] ** WARNING: Using the XFS
S filesystem is strongly recommended with the WiredTiger storage engine
2022-12-17T09:11:38.101-0500 I STORAGE [initandlisten] ** See http://d
ochub.mongodb.org/core/prodnotes-filesystem
2022-12-17T09:11:38.923-0500 I CONTROL [initandlisten]
2022-12-17T09:11:38.923-0500 I CONTROL [initandlisten] ** WARNING: Access contr
ol is not enabled for the database.
2022-12-17T09:11:38.923-0500 I CONTROL [initandlisten] ** Read and wri
```

Hình 58. Thiết lập và khởi động Arbiter

- Bước 5: Bật MongoDB Shell ở máy Primary, thực hiện Arbiter vào cụm phân tán bằng lệnh: `rs.addArb("192.168.1.22:27018")`

```
MongoDB Enterprise g13demo:PRIMARY>
MongoDB Enterprise g13demo:PRIMARY> rs.addArb("192.168.1.22:27018")
{
  "ok" : 1,
  "operationTime" : Timestamp(1671286912, 1),
  "$clusterTime" : {
    "clusterTime" : Timestamp(1671286912, 1),
    "signature" : {
      "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"),
      "keyId" : NumberLong(0)
    }
  }
}
```

Hình 59. Add Arbiter vào liên kết phân tán

- Bước 6: Chờ một vài giây, sau đó ấn Enter ở cả 3 máy để thấy thành quả hoặc thực lệnh `rs.status()` để kiểm tra.

```

root@tphcm:~# mongo
MongoDB shell version v4.0.28
connecting to: mongodb://127.0.0.1:27017/?gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("bce1c4ea-alld-43c0-91b1-38a899bb68b6") }
MongoDB server version: 4.0.28
Server has startup warnings:
2022-12-17T05:10:58.941-0500 I STORAGE [initandlisten]
2022-12-17T05:10:58.941-0500 I STORAGE [initandlisten] ** WARNING: Using the XFS
em is strongly recommended with the WiredTiger storage engine
2022-12-17T05:10:58.941-0500 I STORAGE [initandlisten] ** See http://doch
odb.org/core/prodnotes-filesystem
2022-12-17T05:10:59.980-0500 I CONTROL [initandlisten]
2022-12-17T05:10:59.980-0500 I CONTROL [initandlisten] ** WARNING: Access control
enabled for the database.
2022-12-17T05:10:59.980-0500 I CONTROL [initandlisten] ** Read and write
to data and configuration is unrestricted.
2022-12-17T05:10:59.980-0500 I CONTROL [initandlisten]
MongoDB Enterprise g13demo:PRIMARY>
MongoDB Enterprise g13demo:PRIMARY>

root@hanai:~# mongo
MongoDB shell version v4.0.28
connecting to: mongodb://127.0.0.1:27017/?gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("7909b2ae-cf53-4fcb-8753-5719") }
MongoDB server version: 4.0.28
Server has startup warnings:
2022-12-17T05:11:06.761-0500 I STORAGE [initandlisten]
2022-12-17T05:11:06.761-0500 I STORAGE [initandlisten] ** WARNING: U
system is strongly recommended with the WiredTiger storage engine
2022-12-17T05:11:06.761-0500 I STORAGE [initandlisten] ** S
ongodb.org/core/prodnotes-filesystem
2022-12-17T05:11:07.866-0500 I CONTROL [initandlisten]
2022-12-17T05:11:07.866-0500 I CONTROL [initandlisten] ** WARNING: A
ot enabled for the database.
2022-12-17T05:11:07.866-0500 I CONTROL [initandlisten] ** R
ss to data and configuration is unrestricted.
2022-12-17T05:11:07.866-0500 I CONTROL [initandlisten]
MongoDB Enterprise g13demo:SECONDARY>
MongoDB Enterprise g13demo:SECONDARY>

root@hanai:~# mongo --port 27018
MongoDB shell version v4.0.28
connecting to: mongodb://127.0.0.1:27018/?gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("5a5aae25-672e-40c8-9521-175ee1546271") }
MongoDB server version: 4.0.28
Server has startup warnings:
2022-12-17T09:11:38.101-0500 I STORAGE [initandlisten]
2022-12-17T09:11:38.101-0500 I STORAGE [initandlisten] ** WARNING: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine
2022-12-17T09:11:38.101-0500 I STORAGE [initandlisten] ** See http://dochub.mongodb.org/core/prodnotes-filesystem
2022-12-17T09:11:38.923-0500 I CONTROL [initandlisten]
2022-12-17T09:11:38.923-0500 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2022-12-17T09:11:38.923-0500 I CONTROL [initandlisten] ** Read and write access to data and configuration is unrestricted.
2022-12-17T09:11:38.923-0500 I CONTROL [initandlisten] ** WARNING: You are running this process as the root user, which is not recommended.
2022-12-17T09:11:38.923-0500 I CONTROL [initandlisten]
MongoDB Enterprise g13demo:ARBITER>
MongoDB Enterprise g13demo:ARBITER>

```

Hình 60. Cụm máy phân tán hoàn chỉnh theo cơ chế Replica Set Member

IV. THỰC NGHIỆM MÔ PHỎNG PHÂN TÁN

1. Mô tả bài toán

Chúng tôi giả định bài toán như sau:

Một công ty ABC có hai chi nhánh lớn ở Thành phố Hồ Chí Minh và Hà Nội. Trong đó Thành phố Hồ Chí Minh là trụ sở. Trụ sở ở Thành phố Hồ Chí Minh lưu trữ toàn bộ thông tin khách hàng, có toàn bộ quyền thao tác lên cơ sở dữ liệu. Chi nhánh ở Hà Nội có quyền truy xuất đến cơ sở dữ liệu để phục vụ cho việc quản lý.

Cho biết cơ sở dữ liệu quan hệ toàn cục của công ty gồm thông tin của khách hàng như sau:

QLKH(id, name, diachi)

Tên từ: Mỗi khách hàng có một ID duy nhất, tên khách hàng, địa chỉ khách hàng.

2. Cấu trúc dữ liệu sử dụng

name	diachi
Phuong Thao	TP.HCM
Thanh Nhan	TP.HCM
Phuc Ngo	TP.HCM

3. Các bước thực nghiệm

3.1. Thực nghiệm truy vấn phân tán

- **Bước 1:** Kiểm tra các Database đang có trên máy trụ sở TP.HCM – **Primary**

```
show dbs
```

```
MongoDB Enterprise gl3demo:PRIMARY> show dbs
admin    0.000GB
config   0.000GB
local    0.000GB
```

Hình 61. Kiểm tra database trên máy trụ sở TP-HCM

→ Chúng ta thấy được trên máy trụ sở hiện chỉ có 3 file cấu hình mặc định của MongoDB.

- **Bước 2:** Tạo database QLKH và collection qlkh - **Primary**

```
use qlkh
```

```
db.createCollection('qlkh')
```

```
MongoDB Enterprise g13demo:PRIMARY> use qlkh
switched to db qlkh
MongoDB Enterprise g13demo:PRIMARY> db.createCollection('qlkh')
{
  "ok" : 1,
  "operationTime" : Timestamp(1671272163, 1),
  "$clusterTime" : {
    "clusterTime" : Timestamp(1671272163, 1),
    "signature" : {
      "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAAA="),
      "keyId" : NumberLong(0)
    }
  }
}
```

Hình 62. Tạo database và collection

- **Bước 3:** Insert dữ liệu - **Primary**

```
db.qlkh.insertMany([
```

```
{
```

```
  name: "Phuong Thao",
```

```
  diachi: "TP.HCM"
```

```
},
```

```
{
```

```
  name: "Thanh Nhan",
```

```
  diachi: "TP.HCM"
```

```
},
```

```
{
```

```
  name: "Phuc Ngo",
```

```
  diachi: "TP.HCM"
```

```
}
```

```
])
```



```

MongoDB Enterprise gl3demo:PRIMARY> db.qlkh.insertMany([
...   {
...     name: "Phuong Thao",
...     diachi: "TP.HCM"
...   },
...   {
...     name: "Thanh Nhan",
...     diachi: "TP.HCM"
...   },
...   {
...     name: "Phuc Ngo",
...     diachi: "TP.HCM"
...   }
... ])
{
  "acknowledged" : true,
  "insertedIds" : [
    ObjectId("639d96f73268e2aac66fedbf"),
    ObjectId("639d96f73268e2aac66fedc0"),
    ObjectId("639d96f73268e2aac66fedc1")
  ]
}
MongoDB Enterprise gl3demo:PRIMARY>

```

Hình 63. Insert Data vào Database

→ Chúng tôi không thêm vào dữ liệu cho cột ID vì MongoDB sẽ tự thực hiện đặt ID mặc định.

- **Bước 4:** Kiểm tra lại Collection và dữ liệu đã nhập vào - **Primary**

```

MongoDB Enterprise gl3demo:PRIMARY> show collections
qlkh
MongoDB Enterprise gl3demo:PRIMARY> db.qlkh.find().pretty()
{
  "_id" : ObjectId("639d96f73268e2aac66fedbf"),
  "name" : "Phuong Thao",
  "diachi" : "TP.HCM"
}
{
  "_id" : ObjectId("639d96f73268e2aac66fedc0"),
  "name" : "Thanh Nhan",
  "diachi" : "TP.HCM"
}
{
  "_id" : ObjectId("639d96f73268e2aac66fedc1"),
  "name" : "Phuc Ngo",
  "diachi" : "TP.HCM"
}
MongoDB Enterprise gl3demo:PRIMARY>

```

Hình 64. Kiểm tra collection và data

→ Chúng ta thấy được trong database QLKH đang có 1 collection qlkh và thông tin các khách hàng có trong collections.

- **Bước 5:** Thực hiện cho phép đọc và kiểm tra database trên chi nhánh Hà Nội – **Secondary**

```
rs.secondaryOk()
```

```
show dbs
```

```
MongoDB Enterprise gl3demo:SECONDARY> rs.secondaryOk()
MongoDB Enterprise gl3demo:SECONDARY> show dbs
admin      0.000GB
config     0.000GB
local      0.000GB
qlkh       0.000GB
MongoDB Enterprise gl3demo:SECONDARY>
```

Hình 65. Cho phép đọc và kiểm tra Database ở chi nhánh Hà Nội

→ Chúng ta thấy được ở chi nhánh Hà Nội đã xuất hiện Database QLKH.

- **Bước 6:** Thực hiện kiểm tra collection và truy vấn phân tán – **Secondary**

```
MongoDB Enterprise gl3demo:SECONDARY> use qlkh
switched to db qlkh
MongoDB Enterprise gl3demo:SECONDARY> show collections
qlkh
MongoDB Enterprise gl3demo:SECONDARY>
MongoDB Enterprise gl3demo:SECONDARY> qlkh
2022-12-17T05:18:17.522-0500 E QUERY [js] ReferenceError: qlkh is not defined :
@(shell):1:1
MongoDB Enterprise gl3demo:SECONDARY> db.qlkh.find().pretty()
{
  "_id" : ObjectId("639d96f73268e2aac66fedc0"),
  "name" : "Thanh Nhan",
  "diachi" : "TP.HCM"
}
{
  "_id" : ObjectId("639d96f73268e2aac66fedbf"),
  "name" : "Phuong Thao",
  "diachi" : "TP.HCM"
}
{
  "_id" : ObjectId("639d96f73268e2aac66fedc1"),
  "name" : "Phuc Ngo",
  "diachi" : "TP.HCM"
}
MongoDB Enterprise gl3demo:SECONDARY>
```

Hình 66. Truy vấn phân tán ở chi nhánh Hà Nội.

→ Chúng ta thấy được Database QLKH có 1 collection qlkh và data có trong collection qlkh.

- **Bước 7:** Thực hiện truy vấn phân tán có điều kiện – **Secondary**

➤ Kiểm tra các khách hàng có **diachi** = “TP.HCM”.

```
db.qlkh.find( { diachi: "TP.HCM" } ).pretty()
```

```
MongoDB Enterprise g13demo:SECONDARY> db.qlkh.find( { diachi: "TP.HCM" } ).pretty()
{
  "_id" : ObjectId("639d96f73268e2aac66fedc0"),
  "name" : "Thanh Nhan",
  "diachi" : "TP.HCM"
}
{
  "_id" : ObjectId("639d96f73268e2aac66fedbf"),
  "name" : "Phuong Thao",
  "diachi" : "TP.HCM"
}
{
  "_id" : ObjectId("639d96f73268e2aac66fedc1"),
  "name" : "Phuc Ngo",
  "diachi" : "TP.HCM"
}
```

Hình 67. Truy vấn phân tán có điều kiện – diachi = “TP.HCM”

→ Chúng ta thấy được các khách hàng có **diachi** = “TP.HCM”.

➤ Kiểm tra các khách hàng có **diachi** = “Ha Noi”.

```
db.qlkh.find( { diachi: "Ha Noi" } ).pretty()
```

```
MongoDB Enterprise g13demo:SECONDARY> db.qlkh.find( { diachi: "Ha Noi" } ).pretty()
MongoDB Enterprise g13demo:SECONDARY> []
```

Hình 68. Truy vấn phân tán có điều kiện – diachi = “Ha Noi”

→ Chúng ta thấy được không có bất cứ khách hàng nào có **diachi** = “Ha Noi”.

- **Bước 8:** Thực hiện thêm dữ liệu khách hàng có **diachi** = “Ha Noi” – **Primary**

name	diachi
Phuong Linh	Ha Noi

```
db.qlkh.insertOne(
{
  name: "Phuong Linh",
  diachi: "Ha Noi"
})
```



```
MongoDB Enterprise g13demo:PRIMARY> db.qlkh.insertOne(
...   {
...     name: "Phuong Linh",
...     diachi: "Ha Noi"
...   }
... )
{
  "acknowledged" : true,
  "insertedId" : ObjectId("639d97c43268e2aac66fedc2")
}
MongoDB Enterprise g13demo:PRIMARY> █
```

Hình 69. Thêm dữ liệu khách hàng có diachi = “Ha Noi” trên máy trụ sở

- **Bước 9:** Thực hiện truy vấn dữ liệu khách hàng có diachi = “Ha Noi” – Secondary

```
MongoDB Enterprise g13demo:SECONDARY> db.qlkh.find( { diachi: "Ha Noi" } ).pretty()
MongoDB Enterprise g13demo:SECONDARY> db.qlkh.find( { diachi: "Ha Noi" } ).pretty()
{
  "_id" : ObjectId("639d97c43268e2aac66fedc2"),
  "name" : "Phuong Linh",
  "diachi" : "Ha Noi"
}
MongoDB Enterprise g13demo:SECONDARY> db.qlkh.find().pretty()
{
  "_id" : ObjectId("639d96f73268e2aac66fedc0"),
  "name" : "Thanh Nhan",
  "diachi" : "TP.HCM"
}
{
  "_id" : ObjectId("639d96f73268e2aac66fedbf"),
  "name" : "Phuong Thao",
  "diachi" : "TP.HCM"
}
{
  "_id" : ObjectId("639d96f73268e2aac66fedc1"),
  "name" : "Phuc Ngo",
  "diachi" : "TP.HCM"
}
{
  "_id" : ObjectId("639d97c43268e2aac66fedc2"),
  "name" : "Phuong Linh",
  "diachi" : "Ha Noi"
}
MongoDB Enterprise g13demo:SECONDARY> █
```

Hình 70. Truy vấn dữ liệu khách hàng có diachi = “Ha Noi” trên chi nhánh Hà Nội

→ Chúng ta thấy được dữ liệu vừa được thêm vào từ trụ sở TP.HCM thì ngay lập tức ở chi nhánh Hà Nội đã truy xuất được dữ liệu.

- **Bước 10:** Kiểm tra quyền của role Secondary có đang đúng với cơ chế Replica Set Member hay không? – **Secondary**

```

MongoDB Enterprise g13demo:SECONDARY> db.qlkh.insertOne(
...   {
...     name: "Nguyet",
...     diachi: "Ha Noi"
...   }
... )
2022-12-17T08:50:39.032-0500 E QUERY    [js] WriteCommandError: not master :
WriteCommandError({
  "operationTime" : Timestamp(1671285032, 1),
  "ok" : 0,
  "errmsg" : "not master",
  "code" : 10107,
  "codeName" : "NotMaster",
  "$clusterTime" : {
    "clusterTime" : Timestamp(1671285032, 1),
    "signature" : {
      "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAAA="),
      "keyId" : NumberLong(0)
    }
  }
})
WriteCommandError@src/mongo/shell/bulk_api.js:420:48
Bulk/executeBatch@src/mongo/shell/bulk_api.js:902:1
Bulk/this.execute@src/mongo/shell/bulk_api.js:1150:21
DBCollection.prototype.insertOne@src/mongo/shell/crud_api.js:252:9
@(shell):1:1
MongoDB Enterprise g13demo:SECONDARY> █

```

Hình 71. Insert dữ liệu trên máy Secondary

→ Chúng ta thấy được khi insert dữ liệu từ máy Secondary sẽ bị báo lỗi không có quyền thực hiện do không phải Master (hay là máy Primary).

=> Sau các bước trên chúng ta đã xây dựng được cụm máy phân tán và phân tán được dữ liệu.

3.2. Mô tả thực nghiệm chức năng Arbiter trong cơ chế phân tán Replica Set Member.

- Arbiter không có quyền truy xuất phân tán cũng như các quyền khác đến cơ sở dữ liệu

```
2022-12-17T09:11:38.101-0500 I STORAGE [initandlisten] ** See http://
2022-12-17T09:11:38.923-0500 I CONTROL [initandlisten]
2022-12-17T09:11:38.923-0500 I CONTROL [initandlisten] ** WARNING: Access control
2022-12-17T09:11:38.923-0500 I CONTROL [initandlisten] ** Read and write
2022-12-17T09:11:38.923-0500 I CONTROL [initandlisten] ** WARNING: You are running
2022-12-17T09:11:38.923-0500 I CONTROL [initandlisten]
MongoDB Enterprise gl3demo:ARBITER> rs.secondaryOk()
MongoDB Enterprise gl3demo:ARBITER> show dbs
2022-12-17T09:23:57.347-0500 E QUERY [js] Error: listDatabases failed: {
  "ok" : 0,
  "errmsg" : "node is not in primary or recovering state",
  "code" : 13436,
  "codeName" : "NotMasterOrSecondary"
} :
_getErrorWithCode@src/mongo/shell/utils.js:25:13
Mongo.prototype.getDBs@src/mongo/shell/mongo.js:151:1
shellHelper.show@src/mongo/shell/utils.js:882:13
shellHelper@src/mongo/shell/utils.js:766:15
@(shellhelp2):1:1
MongoDB Enterprise gl3demo:ARBITER>
```

Hình 72. Truy xuất phân tán với role Arbiter

- Khi xảy ra trục trặc hoặc xảy ra trường hợp từ chức, thì trong khi role Primary có thể từ chức và giáng Secondary và role Secondary có thể nhận chức thành Primary thì Arbiter chỉ có thể là Arbiter.

```
Implicit session: session { "id" : UUID("bc4c4ea-af1d-43c0-91b1-38a889bb68b6") } root@hanoi:~# mongo
MongoDB server version: 4.0.28
Server has startup warnings:
2022-12-17T05:10:58.941-0500 I STORAGE [initandlisten]
2022-12-17T05:10:58.941-0500 I STORAGE [initandlisten] ** WARNING: Using the XFS
em is strongly recommended with the WiredTiger storage engine
2022-12-17T05:10:58.941-0500 I STORAGE [initandlisten] ** See http://doc
odb.org/core/prodnotes-filesystem
2022-12-17T05:10:59.980-0500 I CONTROL [initandlisten]
2022-12-17T05:10:59.980-0500 I CONTROL [initandlisten] ** WARNING: Access control
enabled for the database.
2022-12-17T05:10:59.980-0500 I CONTROL [initandlisten] ** Read and write
to data and configuration is unrestricted.
2022-12-17T05:10:59.980-0500 I CONTROL [initandlisten]
MongoDB Enterprise gl3demo:PRIMARY> rs.stepdown()
2022-12-17T09:26:56.947-0500 I NETWORK [js] DBClientConnection failed to receive
from 127.0.0.1:27017 - HostUnreachable: Connection closed by peer
2022-12-17T09:26:56.961-0500 E QUERY [js] Error: error doing query: failed: net
or while attempting to run command 'replSetStepDown' on host '127.0.0.1:27017' :
DB.prototype.runCommand@src/mongo/shell/db.js:170:1
DB.prototype.adminCommand@src/mongo/shell/db.js:188:16
rs.stepdown@src/mongo/shell/utils.js:1505:12
@(shell):1:1
2022-12-17T09:26:56.963-0500 I NETWORK [js] trying reconnect to 127.0.0.1:27017 fa
2022-12-17T09:26:56.965-0500 I NETWORK [js] reconnect 127.0.0.1:27017 ok
MongoDB Enterprise gl3demo:SECONDARY>
MongoDB Enterprise gl3demo:SECONDARY>
root@hanoi:~# mongo --port 27018
MongoDB shell version v4.0.28
connecting to: mongodb://127.0.0.1:27018/?gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("5a5aae25-672e-40c8-9521-175ee1546271") }
MongoDB server version: 4.0.28
Server has startup warnings:
2022-12-17T09:11:38.101-0500 I STORAGE [initandlisten]
2022-12-17T09:11:38.101-0500 I STORAGE [initandlisten] ** WARNING: Using the XFS
2022-12-17T09:11:38.101-0500 I STORAGE [initandlisten] ** See http://dochub.mongodb.org/core/prodnotes-filesystem
2022-12-17T09:11:38.923-0500 I CONTROL [initandlisten]
2022-12-17T09:11:38.923-0500 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2022-12-17T09:11:38.923-0500 I CONTROL [initandlisten] ** Read and write access to data and configuration is unrestricted.
2022-12-17T09:11:38.923-0500 I CONTROL [initandlisten] ** WARNING: You are running this process as the root user, which is not recommend
2022-12-17T09:11:38.923-0500 I CONTROL [initandlisten]
MongoDB Enterprise gl3demo:ARBITER>
MongoDB Enterprise gl3demo:ARBITER>
```

Hình 73. Arbiter sẽ không bao giờ thay đổi role

- Là trong những yếu tố quyết định Secondary nào sẽ nhận chức thành Primary

→ Arbiter sẽ được thể hiện một cách rõ ràng là có vai trò quyết định Secondary nào sẽ nhận chức thành Primary khi xảy ra sự cố hoặc Primary từ chức, số Secondary đang tham gia bầu chọn có ngang số phiếu bầu là số chẵn. Khi đó Arbiter sẽ có nhiệm vụ là một “Vote Only” để phá vỡ cân cân bằng, và Secondary nào được bầu chọn sẽ có số phiếu bầu vượt trội và nhận chức Primary.

```
"ok": 0,
"errmsg": "not master and slaveOk=false",
"code": 13435,
"codeName": "NotMasterNoSlaveOk",
"$clusterTime": {
  "clusterTime": Timestamp(1671287236, 1),
  "signature": {
    "hash": BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAAA"),
    "keyId": NumberLong(0)
  }
}
} :
_getErrorWithCode@src/mongo/shell/utils.js:25:13
Mongo.prototype.getDBs@src/mongo/shell/mongo.js:151:1
shellHelper.show@src/mongo/shell/utils.js:882:13
shellHelper@src/mongo/shell/utils.js:766:15
@(shellhelp2):1:1
MongoDB Enterprise g13demo:SECONDARY> rs.secondaryOk()
MongoDB Enterprise g13demo:SECONDARY> show dbs
admin 0.000GB
config 0.000GB
local 0.000GB
qlkh 0.000GB
MongoDB Enterprise g13demo:SECONDARY>
MongoDB Enterprise g13demo:SECONDARY>
MongoDB Enterprise g13demo:SECONDARY>
MongoDB Enterprise g13demo:PRIMARY>
MongoDB Enterprise g13demo:PRIMARY>
MongoDB shell version v4.0.28
connecting to: mongod://127.0.0.1:27018/?gssapiServiceName=mongod
Implicit session: session {"id": "UUID("5a5aae25-672e-40c8-9521-175ee1546271")"}
MongoDB server version: 4.0.28
Server has startup warnings:
2022-12-17T09:11:38.101-0500 I STORAGE [initandlisten]
2022-12-17T09:11:38.101-0500 I STORAGE [initandlisten]
2022-12-17T09:11:38.101-0500 I STORAGE [initandlisten]
2022-12-17T09:11:38.923-0500 I CONTROL [initandlisten]
2022-12-17T09:11:38.923-0500 I CONTROL [initandlisten]
2022-12-17T09:11:38.923-0500 I CONTROL [initandlisten]
2022-12-17T09:11:38.923-0500 I CONTROL [initandlisten]
2022-12-17T09:11:38.923-0500 I CONTROL [initandlisten]
MongoDB Enterprise g13demo:ARBITER>
MongoDB Enterprise g13demo:ARBITER>
MongoDB Enterprise g13demo:ARBITER>
MongoDB Enterprise g13demo:ARBITER>
MongoDB Enterprise g13demo:ARBITER>
MongoDB Enterprise g13demo:ARBITER>
```

Hình 74. Sự thay đổi role trong cơ chế replica set member

KẾT LUẬN

Sự phát triển không ngừng của công nghệ thông tin, nhu cầu xã hội đòi hỏi những hệ thống phần mềm có khả năng lưu trữ và có tốc độ xử lý cao với một lượng dữ liệu lớn. Trong báo cáo nhóm đã đề cập đến CSDL cơ bản của NoSQL là MongoDB là một CSDL hướng văn bản, lưu trữ dữ liệu dưới cặp key-value. Để tăng tốc độ xử lý truy vấn, người ta thường sử dụng việc đánh chỉ mục và nhúng các đối tượng trong MongoDB. MongoDB tỏ ra đặc biệt hiệu quả với những dự án mà tỉ lệ lượng dữ liệu ghi vào CSDL lớn hơn lượng đọc.

Ưu điểm MongoDB

MongoDB mang đến cho người dùng khá nhiều lợi ích:

- Linh hoạt trong lưu trữ các kích cỡ dữ liệu khác nhau. Nhờ chúng được lưu dưới dạng JSON nên bạn thoải mái chèn bất kỳ thông tin nào tùy theo nhu cầu sử dụng.
- Tiết kiệm thời gian trong việc kiểm tra sự tương thích về cấu trúc khi thêm, xóa hoặc cập nhật dữ liệu. Nhờ MongoDB không có sự ràng buộc trong một khuôn khổ, quy tắc nhất định nào.
- Bạn dễ dàng mở rộng hệ thống thông qua việc thêm node vào cluster. Cụm các node này đóng vai trò như thư viện chứa các dữ liệu giao tiếp với nhau.
- Tốc độ truy vấn của MongoDB nhanh hơn so với RDBMS do toàn bộ dữ liệu truy vấn đã được ghi đệm lên bộ nhớ RAM. Nhờ thế, những lượt truy vấn sau sẽ được rút ngắn thời gian vì chúng không cần đọc từ ổ cứng.
- Trường dữ liệu “_id” (đại diện cho giá trị duy nhất trong mỗi document) được tự động đánh chỉ mục nên hiệu suất luôn đạt mức cao nhất.

Nhược điểm MongoDB

Bên cạnh các ưu điểm, MongoDB vẫn còn tồn tại một số điểm hạn chế mà bạn cần chú ý khi cài đặt và sử dụng:

- Vì dữ liệu không bị ràng buộc nên trong quá trình sử dụng, bạn cần cẩn thận trong mọi thao tác nhằm tránh xảy ra những điều không mong muốn, làm ảnh hưởng đến dữ liệu.
- Chương trình MongoDB tiêu tốn khá nhiều dung lượng bộ nhớ do dữ liệu được lưu dưới dạng key và value. Bên cạnh đó, một số collection chỉ có sự khác biệt về value nên việc lặp lại key là điều khó tránh khỏi. Điều này dẫn đến thừa dữ liệu.

- Thông thường, thời gian để dữ liệu chuyển đổi từ RAM xuống ổ cứng khoảng 60s nên nguy cơ bị mất dữ liệu nếu xảy ra mất điện là điều có thể xảy ra.

Kết quả đạt được

- Nhóm đã đáp ứng được mục đích và yêu cầu đặt ra:
 - Tìm hiểu và hiểu được các cơ sở dữ liệu MongoDB
 - Tìm kiếm, nắm bắt được kỹ thuật sử dụng kết nối với cơ sở dữ liệu MongoDB và nắm được nguyên lý hoạt động của NoSQL-MongoDB.
- Tuy nhiên, do hạn chế về thời gian nên mức độ chuyên sâu nghiên cứu về MongoDB chưa được chính chu và hoàn thiện một số thứ.
- Hướng nghiên cứu và phát triển của nhóm:
 - Tìm hiểu sâu hơn về nhu cầu người dùng và phát triển và tối ưu hóa hệ thống.
 - Tìm hiểu thêm một số công cụ, các phần mềm ứng dụng để nâng cao giao diện đồ họa đẹp mắt, thân thiện hơn,...

TÀI LIỆU THAM KHẢO

- [1] Trang chủ MongoDB, <https://www.mongodb.com/docs/>
- [2] Kristina Chodorow - Michael Dirolf, MongoDB: The Definitive Guide, O'reilly, 2010.
- [3] Z. Wei-Ping, LI Ming-Xin, H. Chen, "Using MongoDB to Implement Textbook Management System instead of MySQL", IEEE 3rd International Conference on Communication Software and Networks (ICCSN), ISSN 978-1-61284-486, 2011.
- [4] [6] S. Hoberman, "Data Modeling for MongoDB", Publisher by Technics Publications, LLC 2 Lindsley Road Basking Ridge, NJ 07920, USA, ISBN 978-1-935504-70-2, 2014
- [5] Replica Set, <https://www.digitalocean.com/community/tutorials/how-to-configure-a-mongodb-replica-set-on-ubuntu-20-04>
- [6] Tìm hiểu về MongoDB, <https://viblo.asia/p/tim-hieu-ve-mongodb-4P856ajGIY3>