

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA KHOA HỌC VÀ KỸ THUẬT THÔNG TIN



BÁO CÁO ĐỒ ÁN CUỐI KỲ
MÔN QUẢN LÝ THÔNG TIN
Đề tài: Tìm hiểu cơ sở dữ liệu Cassandra

GVHD: **ThS. Tạ Thu Thủy**

Sinh viên thực hiện:

1. Trần Văn Tài

MSSV: 21410048

☞ Tp. Hồ Chí Minh, 12/2021 ☞

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

....., ngày.....tháng.....năm 20...

Người nhận xét

(Ký tên và ghi rõ họ tên)

BẢNG PHÂN CÔNG, ĐÁNH GIÁ THÀNH VIÊN:

Bảng 1: Bảng phân công, đánh giá thành viên

Họ và tên	MSSV	Phân công	Đánh giá
Trần Văn Tài	21420048	1. Tìm hiểu khái niệm NoSQL 2. So sánh giữa SQL và NoSQL 3. Tìm hiểu khái niệm Cassandra 4. Cách cài đặt cassandra 5. Sử dụng cassandra như thế nào	Hoàn thành công việc

MỤC LỤC

Chương 1:	Tổng Quan về NoSQL.....	Error! Bookmark not defined.
Chương 2:	Giới thiệu về Cassandra	Error! Bookmark not defined.7
Chương 3:	Cài đặt cassandra với docker.....	Error! Bookmark not defined.
Chương 4:	Thao tác với cassandra bằng dòng lệnh.....	19
TÀI LIỆU THAM KHẢO.....		22

Chương 1: Tổng quan NoSQL

1.1 NoSQL là gì ?

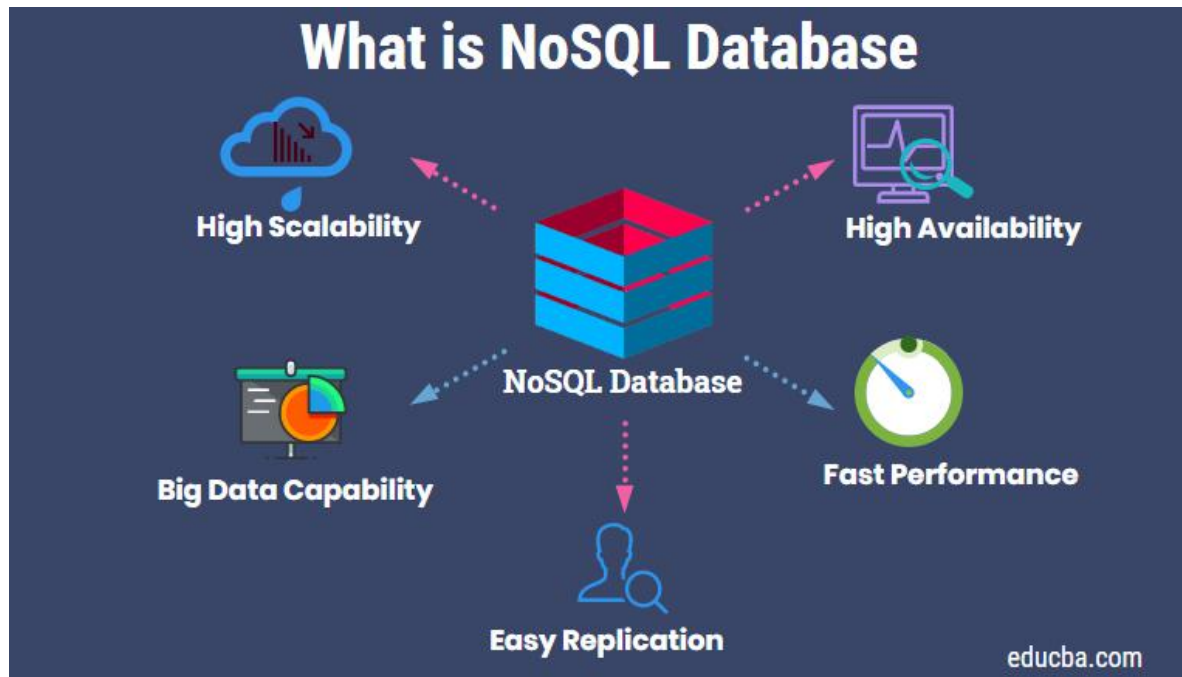


Figure 1 - What is NoSQL ?

[NoSQL. What is NoSQL? | by Jeyasumangala Rasanayagam | Medium](#)

Khi mọi người sử dụng thuật ngữ “cơ sở dữ liệu NoSQL”, họ thường sử dụng nó để chỉ bất kỳ cơ sở dữ liệu không quan hệ nào. Một số người nói rằng thuật ngữ “NoSQL” là viết tắt của “non SQL” trong khi những người khác nói rằng nó là viết tắt của “not only SQL”.

Sao cũng được, hầu hết đều đồng ý rằng cơ sở dữ liệu NoSQL là cơ sở dữ liệu lưu trữ dữ liệu ở định dạng không phải là các bảng và ràng buộc quan hệ bảng quan hệ.

1.2 Lịch sử ra đời của NoSQL

Cơ sở dữ liệu NoSQL xuất hiện vào cuối những năm 2000 khi chi phí dành cho việc lưu trữ giảm đáng kể (chúng ta có các bộ nhớ dung lượng cao hơn, chi phí trên một khối dữ liệu lưu trữ rẻ hơn).

Ví dụ: trước đây 1 ổ cứng **500GB** có giá khoảng 2 triệu vào đầu thập niên 2010, nhưng hiện tại chúng ta có thể mua ổ cứng **2TB(2000GB)** với giá chưa đến 1 triệu đồng

Người ta đã không còn làm công việc là cố gắng tạo ra một mô hình dữ liệu phức tạp để tránh việc bị trùng lặp dữ liệu thứ mà dựa vào các developer (thay vì lưu trữ) là nguồn lao động(cost) chính của việc phát triển phần mềm, do đó, cơ sở dữ liệu NoSQL được tối ưu hóa để nâng cao năng suất của các developer.

Khi chi phí dành cho việc lưu trữ giảm nhanh chóng, lượng dữ liệu mà các ứng dụng cần để lưu trữ và truy vấn tăng lên. Dữ liệu này có đủ hình dạng và kích thước - có cấu trúc, bán cấu trúc và đa hình - và việc xác định trước lược đồ gần như không thể thực hiện được. Cơ sở dữ liệu NoSQL cho phép các nhà phát triển lưu trữ một lượng lớn dữ liệu phi cấu trúc, mang lại cho chúng rất nhiều tính linh hoạt.

Ngoài ra, Với mô hình phát triển phần Agile Manifesto đang ngày càng phổ biến và các kỹ sư phần mềm dần thay đổi tư duy phát triển phần mềm trong đó bản thân phải nhanh chóng thích ứng với các yêu cầu thay đổi của thị trường công nghệ và NoSQL là một trong số đó.

Điện toán đám mây cũng trở nên phổ biến và các nhà phát triển bắt đầu sử dụng các public cloud để lưu trữ các ứng dụng và dữ liệu của họ. Họ muốn khả năng phân phối dữ liệu trên nhiều máy chủ và khu vực để làm cho ứng dụng của họ có khả năng phục hồi, mở rộng theo chiều ngang(**scale out**) thay vì mở rộng theo chiều dọc(**scale up**) và định vị địa lý dữ liệu của họ một cách thông minh.

Cassandra nằm trong số các cơ sở dữ liệu NoSQL đã giải quyết các hạn chế của các công nghệ quản lý dữ liệu trước đây, chẳng hạn như cơ sở dữ liệu SQL.

1.3 Các đặc trưng của NoSQL

Mỗi cơ sở dữ liệu NoSQL có các tính năng độc đáo của riêng nó. At high level, nhiều cơ sở dữ liệu NoSQL có các tính năng sau:

- **Flexible schemas**(lược đồ linh hoạt)
- **Horizontal scaling**(mở rộng theo chiều ngang, hay còn gọi là **Scale Out**)
- **Fast queries due to the data model**(truy vấn nhanh chóng dựa vào model)
- **Ease of use for developers**(thân thiện với các developer)

1.4 Các loại NoSQL

Về cơ bản hiện tại, chúng ta có các loại NoSQL như sau:

Document databases lưu trữ dữ liệu trong tài liệu tương tự như các đối tượng JSON (JavaScript Object Notation). Mỗi tài liệu chứa các cặp trường và giá trị. Các giá trị thường có thể là nhiều loại bao gồm những thứ như chuỗi, số, boolean, mảng hoặc đối tượng.

Key-value là một loại cơ sở dữ liệu đơn giản hơn trong đó mỗi mục chứa các khóa và giá trị.

Wide-column stores lưu trữ dữ liệu trong bảng, hàng và cột động.

Graph databases lưu trữ dữ liệu trong các nút(nodes) và các cạnh(edges). Các nút thường lưu trữ thông tin về người, địa điểm và mọi thứ, trong khi các cạnh lưu trữ thông tin về mối quan hệ giữa các nút.

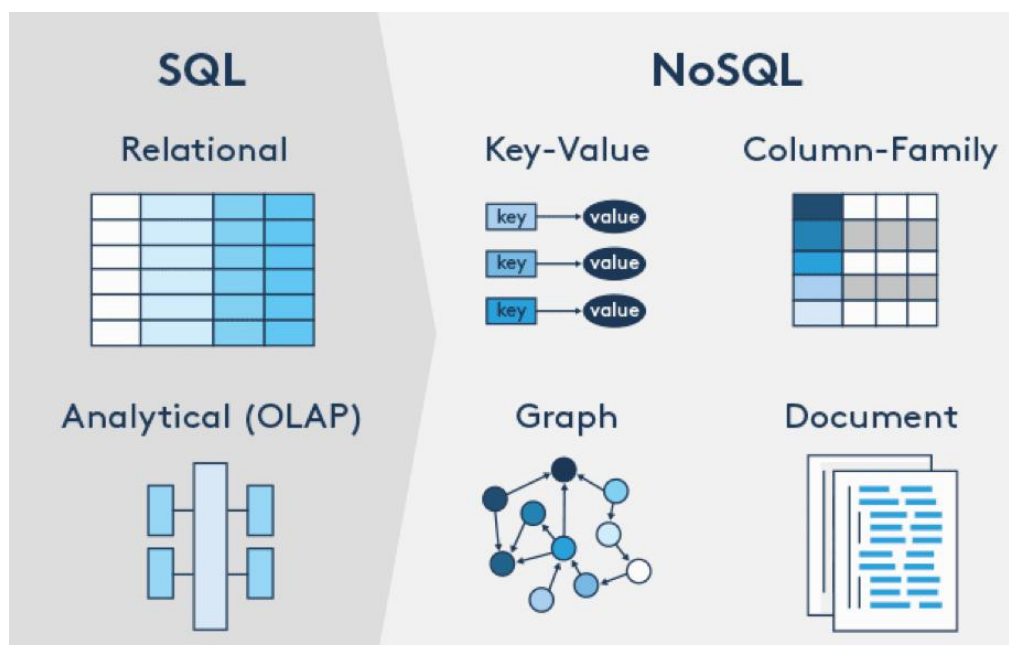


Figure 2 - Các loại model cơ bản của NoSQL hiện tại

[NOSQL Cloud Database Service | All You Need To Know \(k21academy.com\)](https://k21academy.com/nosql-cloud-database-service-all-you-need-to-know)

Chương 2: Giới thiệu về Cassandra

I. Cassandra database là gì và đặc trưng



Cùng với sự bùng nổ của NoSQL thì cassandra ra đời như là 1 giải pháp về lưu trữ, đòi hỏi

Tốc độ và cập nhật tức thời (realtime data)

Có tất cả các đặc trưng của NoSQL

What is Cassandra

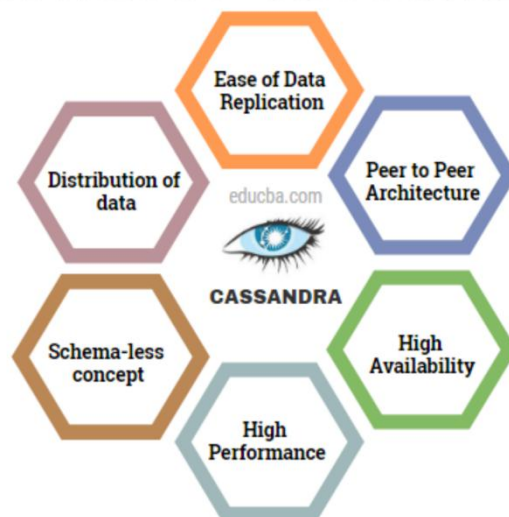


Figure 3 - What is Cassandra ?

[what is Cassandra? | An Comprehensive Guide to Apache Cassandra \(educba.com\)](https://www.educba.com/what-is-cassandra/)

Viết bằng ngôn ngữ java: [Apache Cassandra - Wikipedia](#)

Là một cơ sở dữ liệu phân tán(Distributed database).

Cassandra có khái niệm **keyspace** tương tự trong SQL gọi là **schema database**

Distributed có nghĩa là có thể chạy trên nhiều máy chủ khác nhau trong khi bên ngoài chỉ nhìn thấy nó như là đang truy vấn trên 1 máy tính duy nhất, điều này tận dụng được tối đa sức mạnh phần cứng và tốc độ của chúng.

Cassandra dễ dàng mở rộng kể cả khi đang được sử dụng nhiều và việc distribution dữ liệu cũng tránh được việc mất dữ liệu do bất kỳ lỗi phần cứng nào của data center(cơ chế shard và replica).

Tính “Distributed” của Cassandra là một lợi thế vô vùng mạnh của về nhiều mặt, bao gồm cả kỹ thuật bởi vì một developer có thể tùy chỉnh bằng thông của các truy vấn read/write riêng biệt(ví dụ cho giới hạn tối đa của 1 server là 1000 kết nối đến nó để đọc cùng một thời điểm và giới hạn server A chỉ dùng để đọc, server B chỉ dùng để ghi dữ liệu).

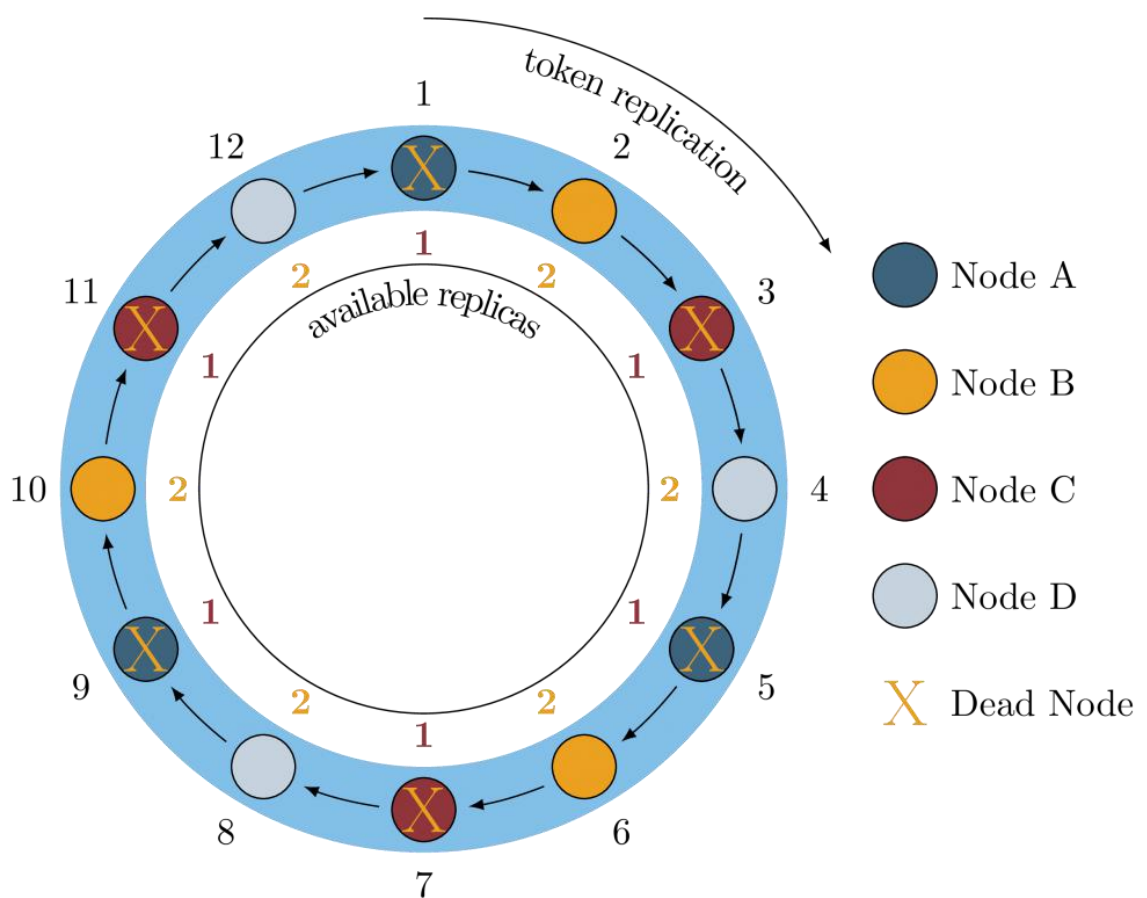


Figure 4 - Cassandra replication architecture scale

[Monitoring Cassandra at Scale \(yelp.com\)](#)

Cơ chế lưu trữ dữ liệu của Cassandra là được lưu trữ trên nhiều nodes, một node đại diện là một cơ sở dữ liệu độc lập (Cassandra). Các node này giao tiếp với nhau thông qua một giao thức được gọi là **gossip**, là một quá trình giao tiếp máy tính ngang hàng với nhau.

Cassandra không có cơ chế master-slave, cơ chế mà mọi tác động tới database đều sẽ đi qua master và node nào là master sẽ chịu trách nhiệm cho việc query data và tổng hợp dữ liệu, từ các node con (slave), nếu một master xảy ra vấn đề thì mất thời gian dài để 1 node có thể trở thành master (mongoDB hiện tại là ~30-40s), trong khoảng thời gian đó database gần như không thể sử dụng, cơ chế hoạt động trong Cassandra là ngang hàng “asynchronous masterless replication allowing low latency operations for all clients”, nơi tất cả các node như là phiên bản độc lập và có khả năng query dữ liệu là như nhau - khả năng chịu lỗi và khả năng phục hồi một cách mạnh mẽ của Cassandra.

Một node có thể có 1 hoặc nhiều máy tính liên kết với nhau, tập hợp nhiều node có thể được tổ chức hợp lý thành cluster, hoặc ring được phân tán trên một hoặc nhiều data center.

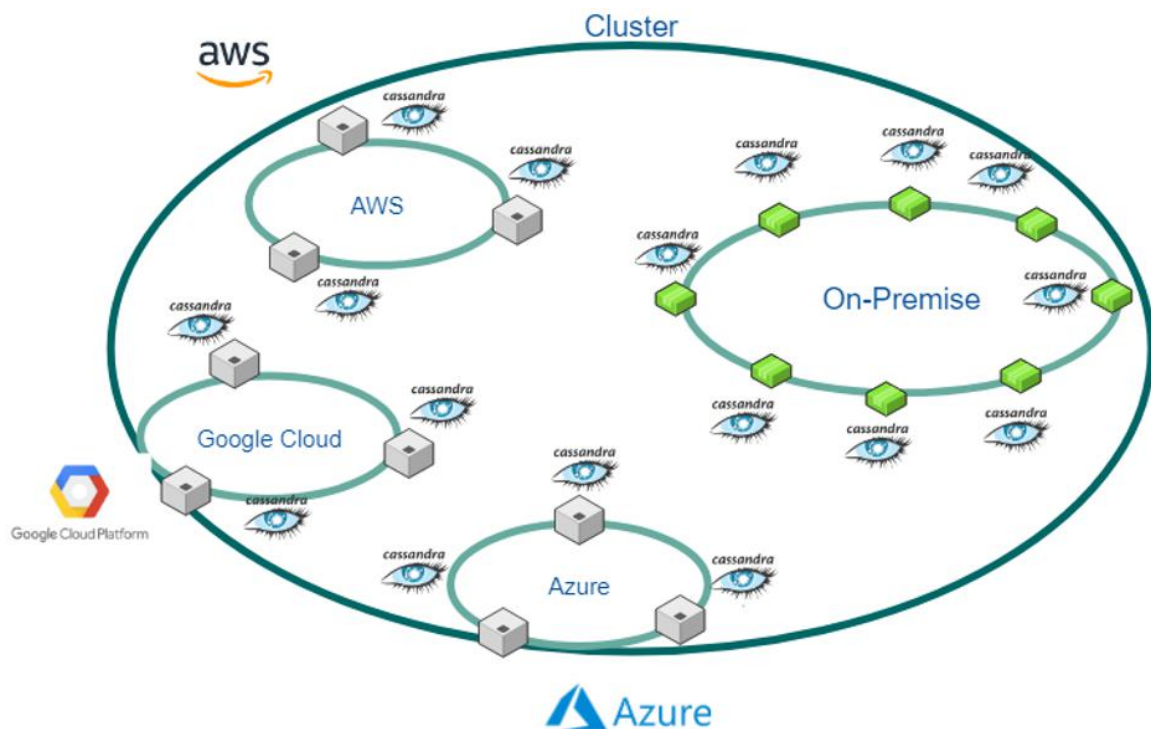


Figure 5 - Thiết kế quản lý cassandra node với mô hình cluster-ring

[5 to 1: An Overview of Apache Cassandra Kubernetes Operators - Container Journal](#)

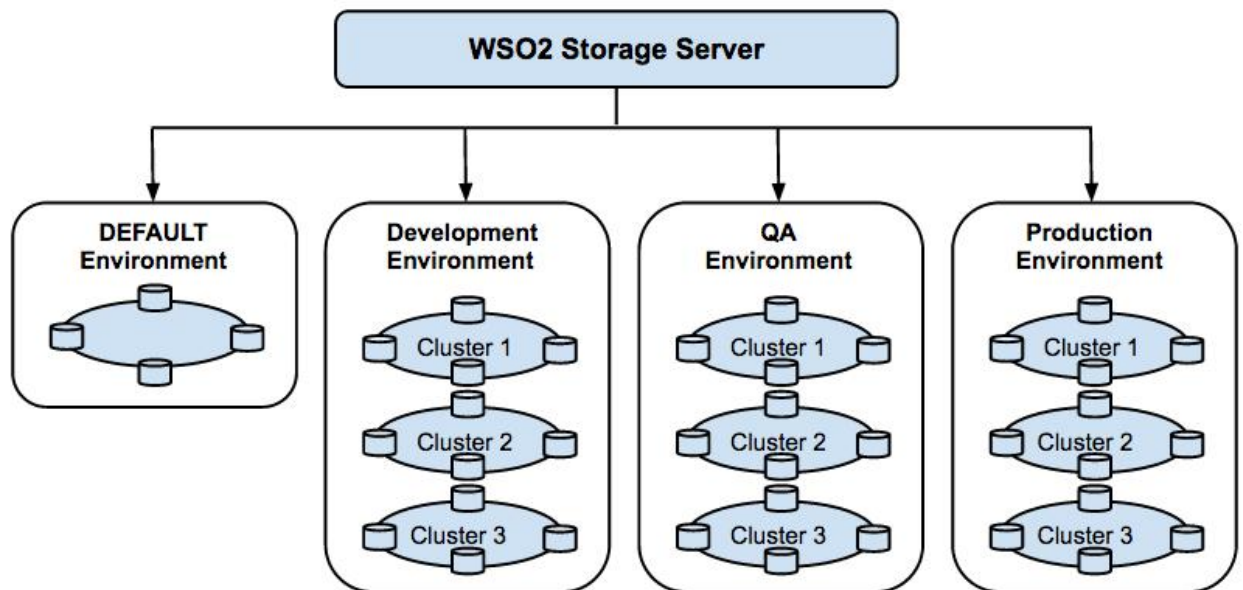


Figure 6 - Thiết kế quản lý cassandra node với mô hình cluster theo vai trò

[Configuring Multiple Cassandra Environments - Storage Server 1.5.0 - WSO2 Documentation](#)

II. Cassandra data types

Native types: kiểu dữ liệu nguyên thủy

Collections types: kiểu dữ liệu tập hợp

User - defined types : kiểu dữ liệu người dùng có thể tự định nghĩa

Tuple types: giống như 1 object trong java, có thể chứa cùng lúc nhiều loại dữ liệu khác nhau trong cùng 1 tuple

Custom types : kiểu dữ liệu đại diện, tức là có thể đặt cho kiểu dữ liệu hiện tại với 1 cái tên khác, không được khuyến khích dùng do không mang lại nhiều lợi ích, **User - defined types** đã có thể thay thế kiểu dữ liệu này

1. Native types

Kiểu dữ liệu	Loại dữ liệu hỗ trợ	Mô tả
ascii	string	Kiểu kí tự ASCII
bigint	integer	Dữ liệu kiểu số, tối đa 64 bit(8 bytes)

blob	blob	Dữ liệu kiểu bytes
boolean	boolean	Kiểu bit, chỉ mang 2 giá trị true(1) hoặc false(0)
counter	integer	Kiểu số, đếm giá trị tăng lên/giảm xuống khi thực hiện câu
date	Integer,string	Kiểu timestamp, String: `1650008820000`, Số: 1650008820000 String `2022-03-15 14:47:00`
decimal	Integer,float	Kiểu số: 123, 123.123
double	Integer,float	Kiểu số thực:, 8 bytes
duration	duration	Kiểu dữ liệu duration, chính xác tới nano seconds
float	Integer,float	Dữ liệu kiểu số thực, 4 bytes dấu chấm động theo chuẩn IEEE-754
inet	string	Kiểu chuỗi, lưu trữ giá trị của IP address: IPv4 hoặc IPv6
int	Integer	Kiểu số nguyên, 4 bytes
smallint	integer	Cũng là kiểu số nguyên nhưng 2 bytes
text	string	Kiểu dữ liệu chuỗi UTF8
time	Integer,string	Số hoặc chuỗi, bao gồm date và time, không bao gồm ngày,tháng,năm
tinyint	integer	Kiểu số nguyên 1 byte
uuid	uuid	UUID
varchar	string	Kiểu chuỗi UTF-8

timestamp	Integer,string	Hỗ trợ date và time(bao gồm, year,month,day)
varint	integer	Số nguyên động

II. Collections Types

map: tập hợp dữ liệu theo dạng **key-value**, implement **hashtable**, mỗi key sẽ là duy nhất trong tập hợp này

sets: một tập hợp theo danh sách, nhưng chưa các phần tử là duy nhất và không có phần tử trùng nhau, các phần tử trong **sets** được sắp xếp sẵn

lists: một tập hợp danh sách, cho phép trùng nhau, các phần tử không được sắp xếp

III. User-Defined Types (UDTs)

Kiểu người dùng tự định nghĩa, về cơ bản chỉ là 1 tập hợp các lệnh mà người dùng cần gom nhóm các lệnh hoặc đặt 1 tên đơn giản, ngắn gọn hơn cho kiểu dữ liệu có sẵn

cú pháp:

```
user_defined_type::=<new name>
<new name>::= <command or data type>
```

Example:

```
user_defined_type::new_defined_name;
new_defined_name::=double;

create table FILM(
    name text,
    rate new_defined_name,
);
```

IV. Tuples

Kiểu dữ liệu tập hợp đặc biệt, nhìn có vẻ giống map nhưng cho phép trùng nhau về key, chứa tối đa 2 phần tử, và 2 phần tử này có thể khác nhau về kiểu dữ liệu

```
tuple<int, text>
```

V. Custom types

Giống như User defined types

Chương 3: Cài đặt Cassandra với docker

Một vài thông tin trước khi cài đặt

Vì cassandra ban đầu được tạo ra cho hệ điều hành linux và không có bản cài đặt sẵn như một phần mềm thường thấy nên chúng ta muốn sử dụng phải cài đặt nó trên một phần mềm mô phỏng linux là docker desktop

Nói sơ qua về docker thì đây có thể xem là 1 dịch vụ máy ảo nơi có thể cài đặt nhiều phần mềm cùng chạy 1 lúc theo dạng image, các service có thể dễ dàng giao tiếp với nhau mà không cần phải cài đặt trên máy tính thật, các service khi cài đặt và khởi chạy gọi là container

MacOS: cài đặt docker desktop for MAC ([Install Docker Desktop on Mac | Docker Documentation](#))

Windows: cài đặt docker desktop for Windows([Install Docker Desktop on Windows | Docker Documentation](#))

Linux: Cài đặt docker service theo hướng dẫn ([Install Docker Engine | Docker Documentation](#))

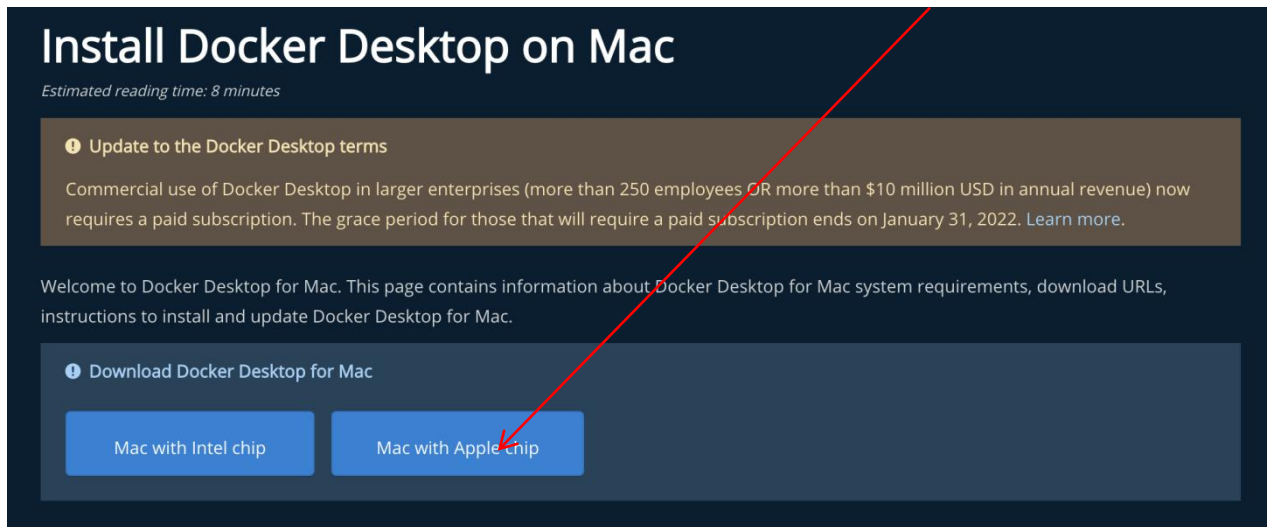
I. Cài đặt docker desktop trên macOS

Trên hệ điều hành macOS, hiện tại vào năm 2020 thì toàn bộ macbook đã chuyển sang dùng kiến trúc chip Arm với tên chip là M1, vì vậy hiện tại trên thị trường máy tính mac tồn tại 2 dòng máy chạy chip intel và M1

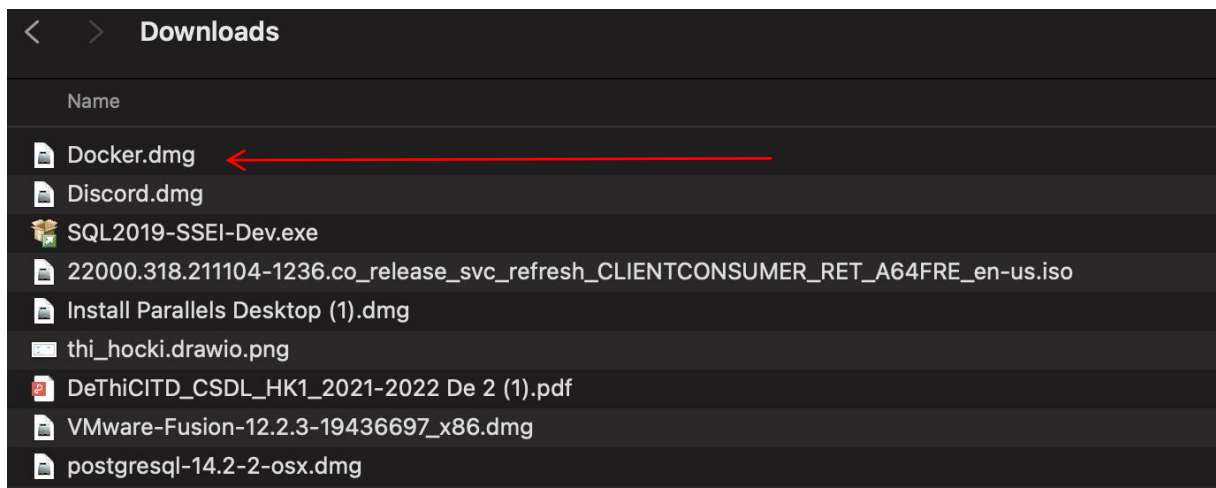
Trước khi tải Docker Desktop cần kiểm tra máy bạn thuộc dòng chip nào để tải bản cài đặt cho phù hợp

Kiểm tra phiên bản của chip máy tính thuộc loại nào

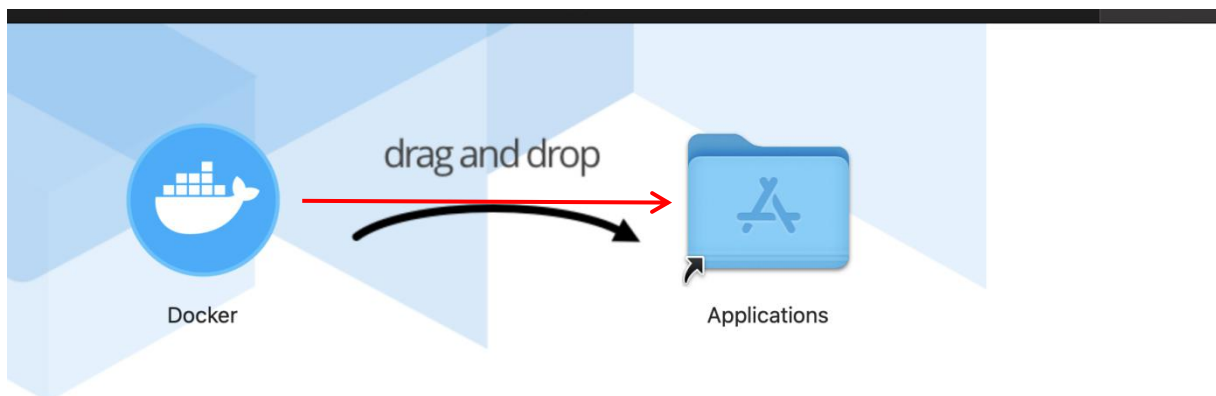




Docker desktop for Mac installation image sau khi tải về

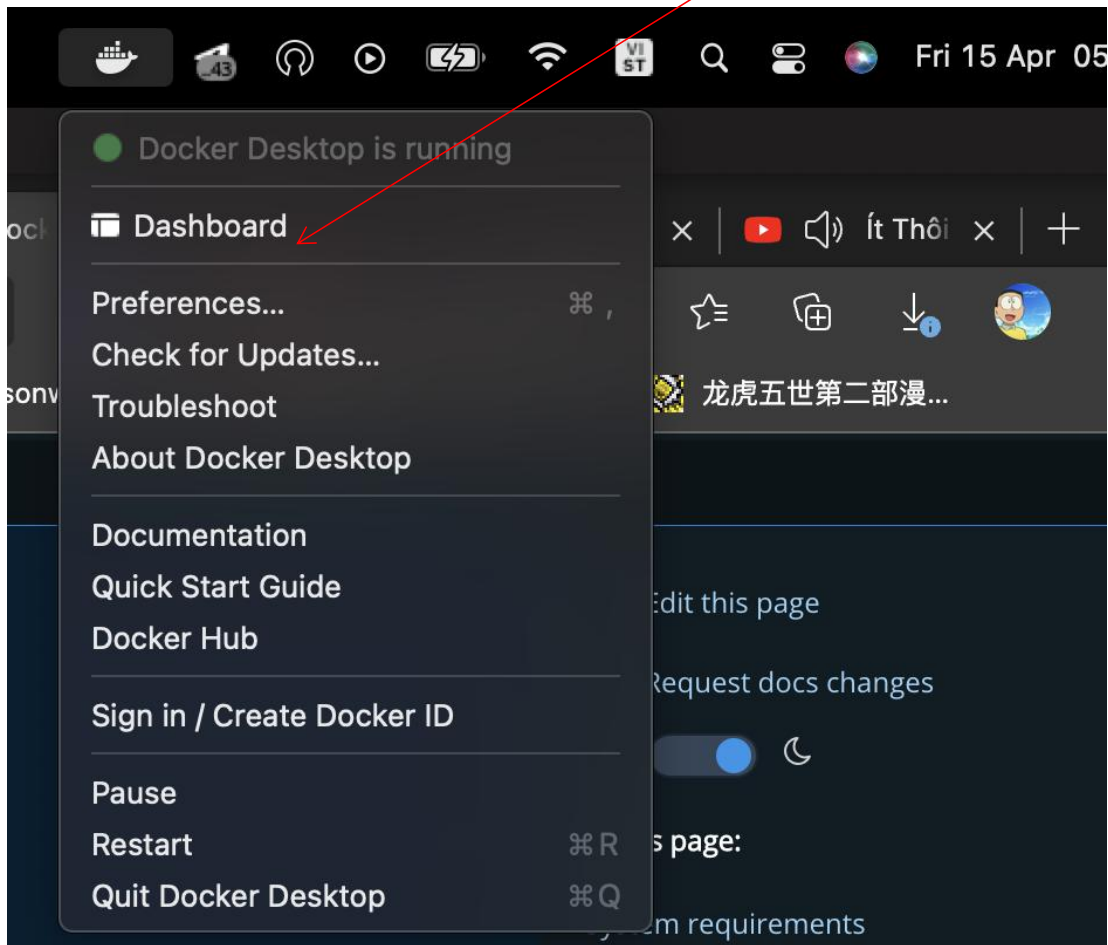


Double-click vào file cài và ta được hình sau, bấm chuột trái và kéo thả icon **Docker** vào folder **Applications** trên hình

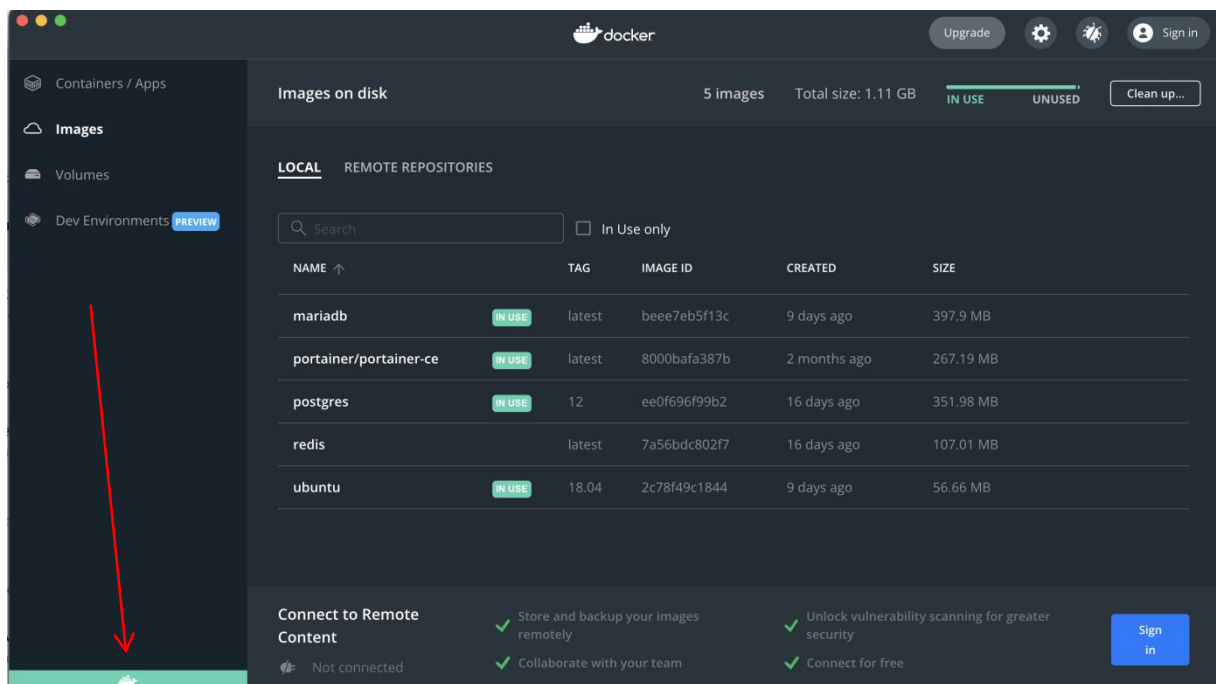


IE103 – Quản lý Thông tin

Sau khi cài xong thì biểu tượng docker đã xuất hiện trên thanh menu bar , click vào và chọn Dashboard



Giao diện dashboard docker desktop



Kiểm tra phiên bản docker bằng lệnh sau: **docker version**

Và ta được thông tin như hình là docker đã được cài đặt hoàn chỉnh và đang chạy thành công trên máy tính

```
> brew docker info
Error: Unknown command: docker
> docker version
Client:
 Cloud integration: v1.0.22
 Version:          20.10.12
 API version:      1.41
 Go version:       go1.16.12
 Git commit:       e91ed57
 Built:            Mon Dec 13 11:46:56 2021
 OS/Arch:          darwin/arm64
 Context:          default
 Experimental:     true

Server: Docker Desktop 4.5.0 (74594)
Engine:
 Version:          20.10.12
 API version:      1.41 (minimum version 1.12)
 Go version:       go1.16.12
 Git commit:       459d0df
 Built:            Mon Dec 13 11:43:07 2021
 OS/Arch:          linux/arm64
 Experimental:     false
containerd:
 Version:          1.4.12
 GitCommit:        7b11cfaabd73bb80907dd23182b9347b4245eb5d
runc:
 Version:          1.0.2
 GitCommit:        v1.0.2-0-g52b36a2
docker-init:
 Version:          0.19.0
 GitCommit:        de40ad0
```

II. Cài đặt Cassandra

Mở terminal và chạy dòng lệnh sau để tiến hành cài đặt cassandra trên docker

docker pull cassandra:latest

```
GitCommit: de40ad0
> docker pull cassandra:latest
latest: Pulling from library/cassandra
185e8a4c1005: Already exists
749a2a96ed3e: Pull complete
fa473dc99209: Downloading [=====] 32.76MB/41.57MB
4a64027b2639: Download complete
2934b62f82b3: Download complete
ea3d3e4f2dd0: Download complete
9e6caaef001b: Download complete
4dc8eb95b6d4: Downloading [=] 1.025MB/49.79MB
9a273c9bef20: Download complete
```

Cài đặt thành công


```

> docker pull cassandra:latest
latest: Pulling from library/cassandra
185e8a4c1005: Already exists
749a2a96ed3e: Pull complete
fa473dc99209: Pull complete
4a64027b2639: Pull complete
2934b62f82b3: Pull complete
ea3d3e4f2dd0: Pull complete
9e6caaef001b: Pull complete
4dc8eb95b6d4: Pull complete
9a273c9bef20: Pull complete
Digest: sha256:3e740efec16311884cec166fe90e0cfdd515b2fc9bd35efc1d870eac254f01a6
Status: Downloaded newer image for cassandra:latest
docker.io/library/cassandra:latest

```

Kiểm tra image cassandra vừa pull về: **docker images**

```

> docker images

```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
cassandra	latest	3079f77c0f57	8 days ago	327MB
mariadb	latest	beee7eb5f13c	8 days ago	398MB
ubuntu	18.04	2c78f49c1844	9 days ago	56.7MB
redis	latest	7a56bdc802f7	2 weeks ago	107MB
postgres	12	ee0f696f99b2	2 weeks ago	352MB
portainer/portainer-ce	latest	8000bafa387b	2 months ago	267MB

Truy cập vào console container bằng command: `docker exec -it <container name code> bash`

```
docker exec -it 81821197a60c bash
```

```

Run a Command in a Running Container
> docker ps
CONTAINER ID   IMAGE
81821197a60c   cassandra
9b0e7014c03b   postgres:12
c99129adad4a   portainer/portainer-ce:latest
> docker exec -it 81821197a60c bash
root@cassandra:/#

```

Vậy là đã hoàn tất việc cài đặt cassandra trên máy tính mac, chương tiếp theo sẽ hướng dẫn việc thao tác và sử dụng cassandra bằng dòng lệnh.

Chương 4: Thao tác với cassandra dòng lệnh

1. Truy cập vào cassandra bằng dòng lệnh sau: `cqlsh localhost 9042`

```
> docker exec -it 81821197a60c bash
root@cassandra:/# cqlsh localhost 9042
Connected to Test Cluster at localhost:9042
[cqlsh 6.0.0 | Cassandra 4.0.3 | CQL spec 3.4.5 | Native protocol v5]
Use HELP for help.
cqlsh>
```

2. Xem thông tin toàn bộ keyspace đang có trên cassandra

`desc keyspace;`

```
(0 rows)
cqlsh:system_auth> desc keyspaces;

system          system_distributed  system_traces  system_virtual_schema
system_auth     system_schema       system_views
cqlsh:system_auth>
```

3. Để sử dụng và truy vấn 1 keyspace: `use <keyspace>`

Ví dụ: `use system;`

4. Xem mô tả của toàn bộ table của keyspace hiện tại

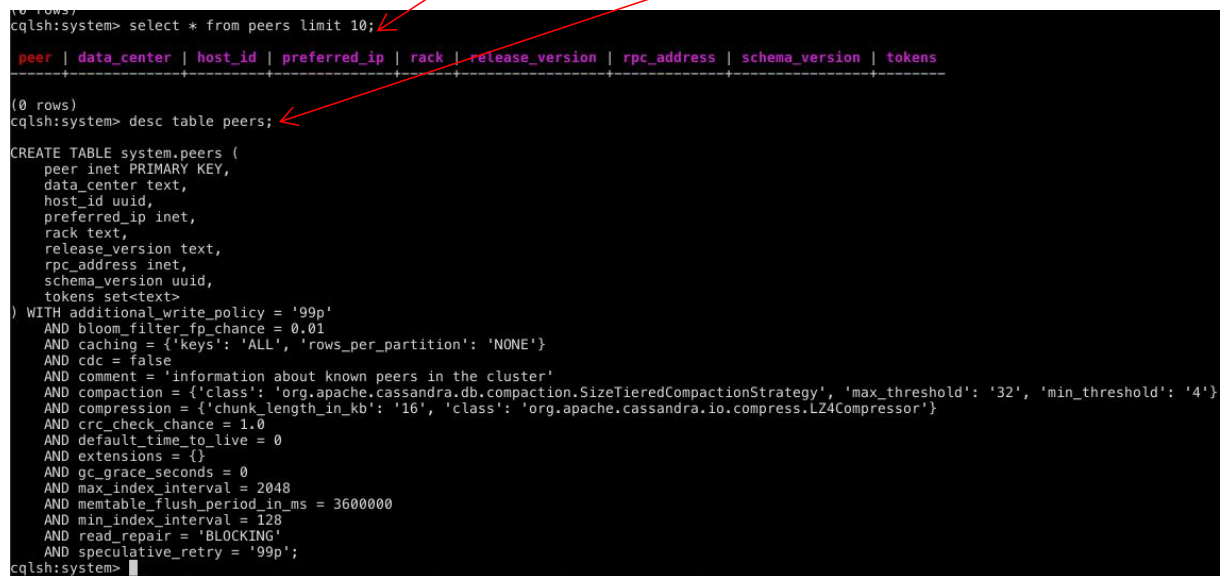
`desc tables;`

```
cqlsh:system_auth> use system;
cqlsh:system> desc tables;

available_ranges      paxos                size_estimates
available_ranges_v2  peer_events          sstable_activity
batches              peer_events_v2       table_estimates
built_views          peers                transferred_ranges
compaction_history   peers_v2             transferred_ranges_v2
"IndexInfo"          prepared_statements  view_builds_in_progress
local                repairs
cqlsh:system> █
```

Query thử trên 1 table *peers*: `select * from peers limit 10;`

Xem toàn bộ thông tin liên quan table cụ thể như kiểu dữ liệu, số column, ràng buộc... bằng lệnh:
`desc table <table name> <==> desc table peers`



```
cqlsh:system> select * from peers limit 10;
peer | data_center | host_id | preferred_ip | rack | release_version | rpc_address | schema_version | tokens
-----+-----+-----+-----+-----+-----+-----+-----+-----
(0 rows)

cqlsh:system> desc table peers;
CREATE TABLE system.peers (
  peer inet PRIMARY KEY,
  data_center text,
  host_id uuid,
  preferred_ip inet,
  rack text,
  release_version text,
  rpc_address inet,
  schema_version uuid,
  tokens set<text>
) WITH additional_write_policy = '99p'
AND bloom_filter_fp_chance = 0.01
AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
AND cdc = false
AND comment = 'information about known peers in the cluster'
AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'max_threshold': '32', 'min_threshold': '4'}
AND compression = {'chunk_length_in_kb': '16', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}
AND crc_check_chance = 1.0
AND default_time_to_live = 0
AND extensions = {}
AND gc_grace_seconds = 0
AND max_index_interval = 2048
AND memtable_flush_period_in_ms = 3600000
AND min_index_interval = 128
AND read_repair = 'BLOCKING'
AND speculative_retry = '99p';
cqlsh:system>
```

5. Tạo keyspace và table trên cassandra

Chúng ta sẽ tạo một keyspace tên **store** bằng câu lệnh dưới đây

```
-- Create a keyspace
```

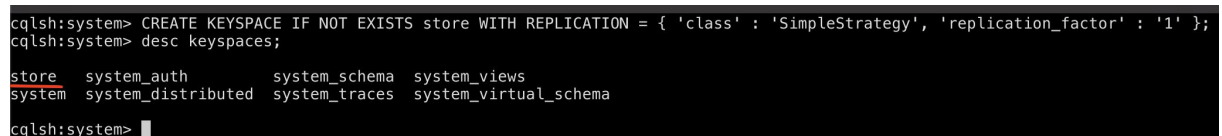
```
CREATE KEYSPACE IF NOT EXISTS store WITH REPLICATION = { 'class' :
'SimpleStrategy', 'replication_factor' : '1' };
```

Chúng ta sẽ tạo một table **shopping_cart** trên keyspace **store** bằng câu lệnh dưới đây

```
-- Create a table
```

```
CREATE TABLE IF NOT EXISTS store.shopping_cart (
userid text PRIMARY KEY,
item_count int,
last_update_timestamp timestamp
);
```

Tạo database



```
cqlsh:system> CREATE KEYSPACE IF NOT EXISTS store WITH REPLICATION = { 'class' : 'SimpleStrategy', 'replication_factor' : '1' };
cqlsh:system> desc keyspaces;
store      system_auth      system_schema    system_views
system     system_distributed system_traces     system_virtual_schema
cqlsh:system>
```

IE103 – Quản lý Thông tin

Chuyển từ 1 database khác sang database store vừa tạo: user store

và tạo table shopping_cart

Sau đó kiểm tra thông tin table trên database vừa tạo bằng dòng lệnh: desc store

```
cqlsh:store> use store;
cqlsh:store> CREATE TABLE IF NOT EXISTS store.shopping_cart ( userid text PRIMARY KEY, item_count int, last_update_timestamp timestamp );
cqlsh:store> desc store;

CREATE KEYSPACE store WITH replication = {'class': 'SimpleStrategy', 'replication_factor': '1'} AND durable_writes = true;

CREATE TABLE store.shopping_cart (
  userid text PRIMARY KEY,
  item_count int,
  last_update_timestamp timestamp
) WITH additional_write_policy = '99p'
AND bloom_filter_fp_chance = 0.01
AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
AND cdc = false,
AND comment = ''
AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'max_threshold': '32', 'min_threshold': '4'}
AND compression = {'chunk_length_in_kb': '16', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}
AND crc_check_chance = 1.0
AND default_time_to_live = 0
AND extensions = {}
AND gc_grace_seconds = 864000
AND max_index_interval = 2048
AND memtable_flush_period_in_ms = 0
AND min_index_interval = 128
AND read_repair = 'BLOCKING'
AND speculative_retry = '99p';
cqlsh:store> █
```

Vì table vừa tạo chưa được insert một dòng dữ liệu nào nên khi query sẽ không thấy bất cứ gì:

```
select * from store limit 10;
```

```
AND speculative_retry = '99p';
cqlsh:store> select * from shopping_cart limit 10;

userid | item_count | last_update_timestamp
-----+-----+-----
(0 rows)
cqlsh:store> █
```

Câu lệnh insert database shopping_cart vừa tạo

```
INSERT INTO store.shopping_cart(userid, item_count,
last_update_timestamp)VALUES ('9876', 2, toTimeStamp(now()));
```

```
INSERT INTO store.shopping_cart(userid, item_count,
last_update_timestamp)VALUES ('1234', 5, toTimeStamp(now()));
```

```
(0 rows)
cqlsh:store> INSERT INTO store.shopping_cart(userid, item_count, last_update_timestamp)VALUES ('9876', 2, toTimeStamp(now()));
cqlsh:store> INSERT INTO store.shopping_cart(userid, item_count, last_update_timestamp)VALUES ('1234', 5, toTimeStamp(now()));
cqlsh:store> select * from shopping_cart limit 10;

userid | item_count | last_update_timestamp
-----+-----+-----
1234   | 5         | 2022-04-16 17:33:20.624000+0000
9876   | 2         | 2022-04-16 17:33:19.554000+0000
(2 rows)
cqlsh:store>
```

sau khi insert và chạy lại câu query, đã có dữ liệu

TÀI LIỆU THAM KHẢO:

- 1/ [Apache Cassandra | Apache Cassandra Documentation](#)
- 2/ [Data Types | Apache Cassandra Documentation](#)
- 3/ [Apache Cassandra - Wikipedia](#)
- 4/ [Configuring Multiple Cassandra Environments - Storage Server 1.5.0 - WSO2 Documentation](#)
- 5/ [what is Cassandra? | An Comprehensive Guide to Apache Cassandra \(educba.com\)](#)
- 6/ [NoSQL. What is NoSQL? | by Jeyasumangala Rasanayagam | Medium](#)
- 7/ [5 to 1: An Overview of Apache Cassandra Kubernetes Operators - Container Journal](#)
- 8/ [Monitoring Cassandra at Scale \(yelp.com\)](#)