1. **GIỚI THIỆU VỀ HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU RETHINKDB**
2. **RETHINKDB LÀ GÌ?**

**RethinkDB** là cơ sở dữ liệu mã nguồn mở cấu trúc JSON đầu tiên được xây dựng cho các ứng dụng và website thực thi theo thời gian thực.

Hệ quản trị **RethinkDB** không sử dụng kiến trúc cơ sở dữ liệu truyền thống bằng cách đưa ra một mô hình truy cập mới rất thú vị, giúp các lập trình viên có thể liên tục cập nhật kết quả truy vấn đến cac ứng dụng.

Với kiến trúc *realime – push* của **RethinkDB**, thời gian cũng như tài nguyên cần thiết để xây dựng các ứng dụng theo thời gian thực. Cùng với đó, kiểu hệ quản trị cơ sở dữ liệu phân tán đem đến khả năng mở rộng cao.

Cùng với đó, **RethinkDB** sử dụng ngôn ngữ truy vấn **ReQL** rất linh hoạt, hoạt động trực quan và dễ sử dụng.

1. **RETHINKDB ĐƯỢC CÁC DOANH NGHIỆP NÀO SỬ DỤNG?**

**RethinkDB** được sử dụng bởi hàng trăm công ty startup lĩnh vực công nghệ, các studios, và 500 công ty Fortune. Một số ví dụ điển hình là:

* **Jive Software** sử dụng **RethinkDB** trong các website hay ứng dụng di động reactive.
* **Pristine.io** và **Narrative Clip** sử dụng **RethinkDB** trong hệ thống lưu trữ đám mây để kết nối các thiết bị.
* **Platzi** và **Workshape.io** sử dụng **RethinkDB** với mục tiêu thực hiện các phân tích theo thời gian thực.
* **NodeCraft** và **GameServerKings** dùng **RethinkDB** trong việc mở rộng thêm các trò chơi (multiplayer games).

**RethinkDB** hiện đã đạt được một cộng đồng sôi động với hơn 100.000 nhà phát triển và hơn 100 cộng tác viên từ khắp nơi trên thế giới.

1. **RETHINKDB ĐƯỢC XÂY DỰNG DỰA TRÊN CÔNG NGHỆ NÀO HIỆN CÓ?**

Việc triển khai kiến trúc realtime-push một cách hiệu quả, các nhà phát triển phải thiết kế lại hầu hết các thành phần cơ sở dữ liệu, bao gồm công cụ thực thi truy vấn, hệ thống phân tán, hay công cụ lưu trữ… Do đó, **RethinkDB** đã được triển khai trong ngôn ngữ C++ ngay từ đầu.

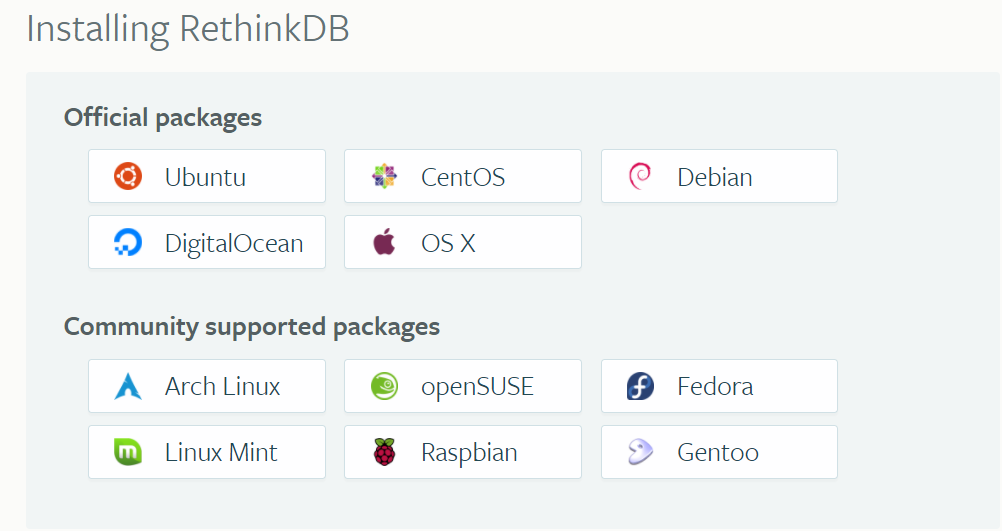
1. **ĐIỂM MẠNH VÀ ĐIỂM YẾU CỦA HỆ QUẢN TRỊ RETHINKDB**

|  |  |
| --- | --- |
| **ĐIỂM MẠNH** | **HẠN CHẾ** |
| * Sử dụng ngôn ngữ truy vấn có tính trực quan cao. * Nhiều phương thức truy vấn liên quan giữa nhiều bảng như join table. * Cập nhật – truy vấn dữ liệu liên tục. * Giao diện quản trị web thân thiện, trực quan và dễ sử dụng. | * Giới hạn cụm / phân đoạn. * Giới hạn bảng / documents. * Giới hạn về khóa. * Giới hạn về kiểu dữ liệu. |

1. **HƯỚNG DẪN SỬ DỤNG HỆ QUẢN TRỊ RETHINKDB**
2. **INSTALLATION**

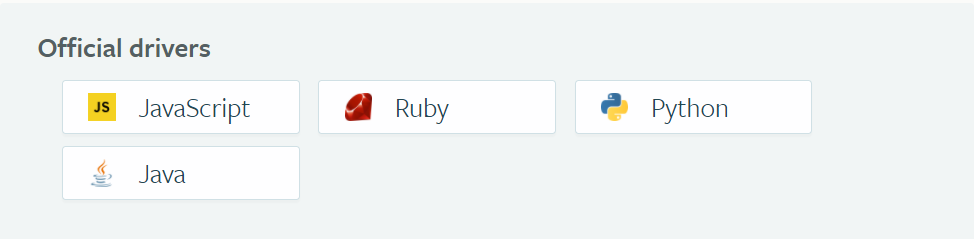
**Bước 1 – CÀI ĐẶT SERVER**

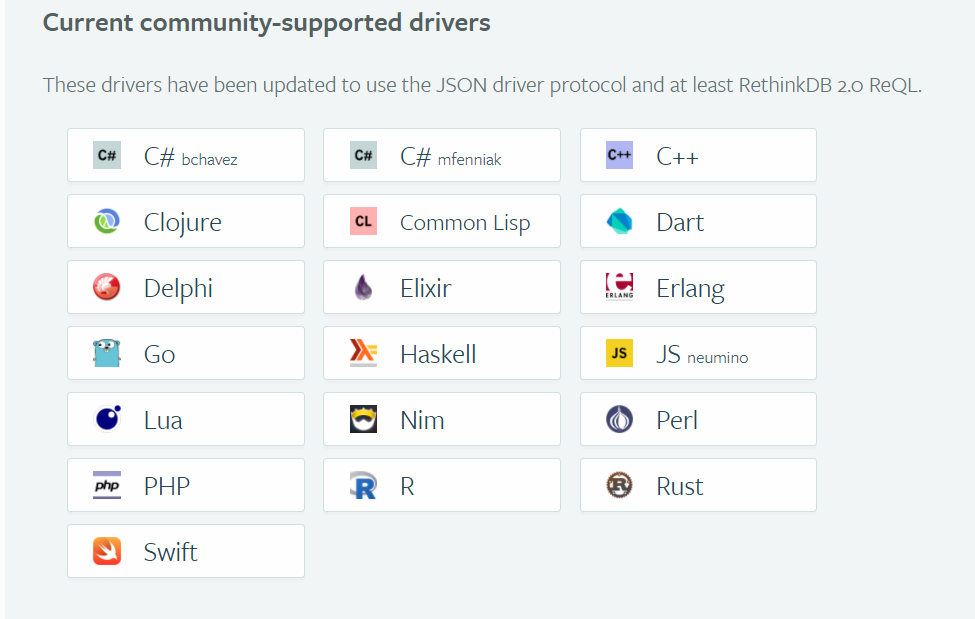
Truy cập vào trang web <https://rethinkdb.com/docs/install/>, sau đó chọn packages phù hợp với hệ điều hành hiện có.



**Bước 2 – CÀI ĐẶT CLIENT DRIVERS**

Truy cập vào trang web <https://rethinkdb.com/docs/install-drivers/> để cài đặt driver tương ứng với ngôn ngữ đang sử dụng.





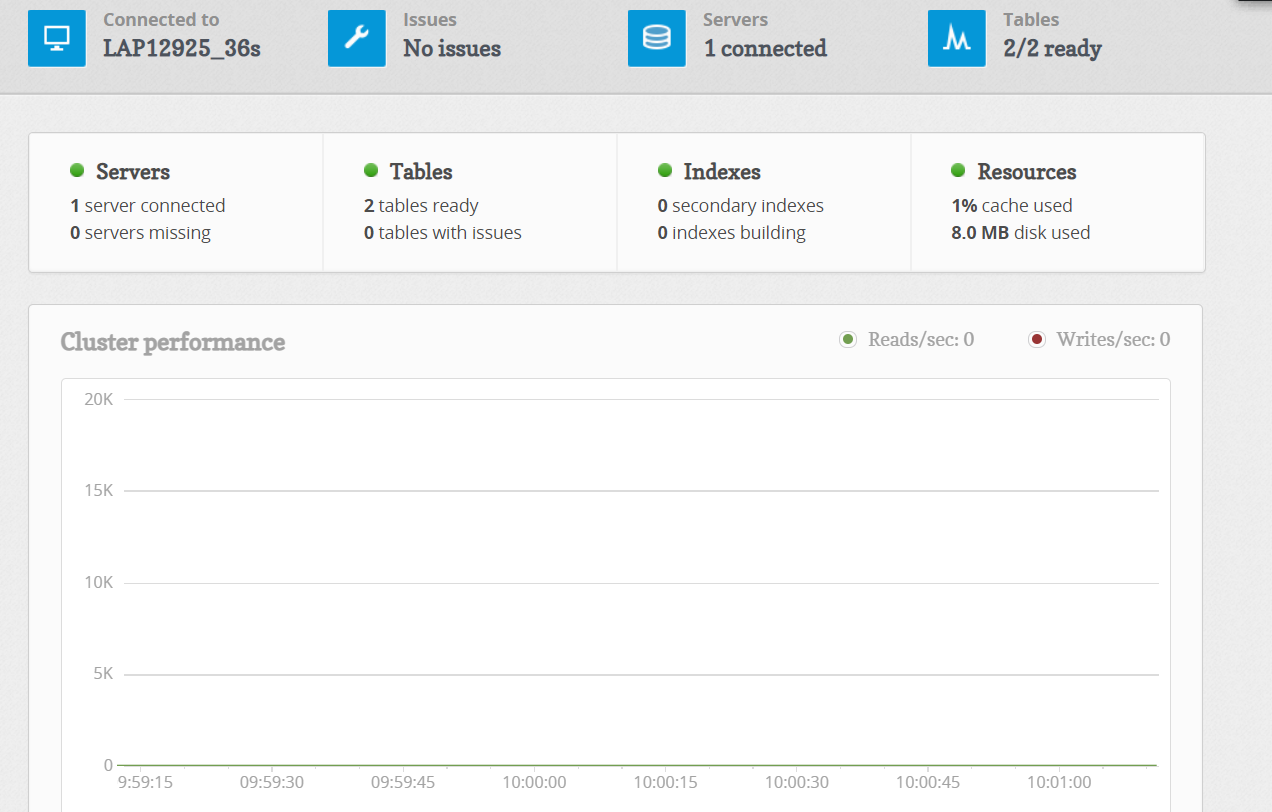
**Bước 3 – KHỞI ĐỘNG SERVER**

Để bắt đầu sử dụng server **RethinkDB**, chạy câu lệnh sau trong terminal:



Các port numbers có thể dùng để truy cập **RethinkDB** gồm:

* Port nội bộ (mặc định **29015**) để truy ca65o đến các node khác trong cùng 1 cluster với node hiện tại.
* Để truy cập vào web interface, truy cập vào localhost:8080, giao diện web sẽ có hình ảnh sau:



1. **CẤU TRÚC DỮ LIỆU CỦA RETHINKDB**

1 **Cluster** có thể có nhiều **Database**.

**CLUSTER**

1 **Database** có thể có nhiều **Table**.

**DATABASE**

**TABLE**

1 **Table** có thể có nhiều **Document**.

**DOCUMENTS**

**Document mẫu trong RethinkDB:**

{

“name”: “Will Riker”,

“Position”: “Commander”,

“height”: 193,

“birthdate”: “Mon Aug 19 2335”

“ships”: [

{“name”: “USS pegasus”},

{“name”: “USS Potemkin”},

{“name”: “USS Enterprise”},

],

…

}

1. **CẤU TRÚC CÂU LỆNH TRUY VẤN THEO REQL**

**THAO TÁC TRÊN DATABASE**

1. **Tạo database:**

r.dbCreate(dbName)

Khởi tạo 1 **database** (một collection các bảng, tương tự như cơ sở dữ liệu quan hệ).

Nếu thành công, câu lệnh sẽ trả về 1 object với 2 fields:

* + **dbs\_created:** luôn là 1.
  + **config\_changes**: một danh sách chứa 1 đối tượng với 2 fields:
    - **old\_val**: luôn là **null**.
    - **new\_val**: giá trị cấu hình mới từ cơ sở dữ liệu.

Nếu database cùng tên đã tồn tại, lệnh sẽ ném lỗi **ReqlRuntimeError**.

*Ví dụ,* tạo một database với tên gọi ‘**superheroes’**:



1. **Xóa database**

r.dbDrop(dbName)

Xóa một database với tất cả các bảng và dữ liệu thuộc database đó. Nếu thành công, ta sẽ nhận được một object với 3 fields:

* + **dbs\_dropped**: luôn là 1.
  + **table\_dropped:** số lượng bảng trong database bị xóa.
  + **config\_changes:** một danh sách các object gồm 2 field:
    - **old\_val:** giá trị gốc của database.
    - **new\_val:** luôn là null.

Nếu database không tồn tại, lệnh sẽ ném lỗi **ReqlRuntimeError**.

*Ví dụ,* ta xóa database có tên ‘**superheroes’**:



1. **Liệt kê tất cả các database:**

r.dbList()

Lệnh sẽ xuất ra danh sách tất cả database trong hệ thống dưới dạng danh sách các chuỗi.

**THAO TÁC TRÊN BẢNG**

1. **Tạo table:**
2. **SO SÁNH RETHINKDB VỚI CÁC HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU KHÁC**

Ở các hệ quản trị CSDL truyền thống, để thực hiện thay đổi, chúng ta cần phải gửi yêu cầu thông qua các câu query và sau đó chờ sự phản hồi ngược lại từ hệ quản trị. Còn với **RethinkDB**, các cập nhật trên database sẽ trực tiếp thay đổi ứng dụng mà không cần polling.

1. **SO SÁNH RETHINKDB VỚI CÁC HỆ THỐNG ĐỒNG BỘ HÓA THỜI GIAN THỰC KHÁC**

**RethinkDB** về cơ bản khác với các API đồng bộ hóa theo thời gian thực khác như *Firebase, PubNub hay Pusher* ở 3 điểm chính.

*Thứ nhất*, các API đồng bộ thời gian thực là các dịch vụ đám mây, còn **RethinkDB** là mã nguồn mở. Mặc dù **RethinkDB** có sẵn trên Cloud thông qua các đối tác như Compose.io và Amazon AWS, nhưng nó cũng có thể được triển khai trong cơ sở hạ tầng riêng mà không bị hạn chế.

*Thứ hai,* các API đồng bộ hóa thời gian thực bị giới hạn trong việc đồng bộ hóa tài liệu cũng như khả năng truy vấn dữ liệu, trong khi **RethinkDB** là một hệ thống cơ sở dữ liệu chung. Ở **RethinkDB**, có thể chạy truy vấn tùy ý bao gồm kết hợp bảng, truy vấn con, truy vấn không gian địa lý,…

*Cuối cùng,* các API đồng bộ hóa thời gian thực được thiết kế để truy cập trực tiếp từ trình duyệt. Điều này có thể hỗ trợ thiết lập dễ dàng để chạy ứng dụng cơ bản, nhưng hạn chế tính linh hoạt khi ứng dụng mở rộng. Trong khi đó, **RethinkDB** được thiết kế để có thể truy cập từ một máy chủ ứng dụng, giống như một cơ sở dữ liệu truyền thống. Do đó, cơ sở dữ liệu sẽ mở rộng rất dễ dàng khi ứng dụng trở nên phức tạp hơn.

1. **RETHINKDB KHÁC VỚI MONGODB NHƯ THẾ NÀO?**

**RethinkDB** dựa trên một kiến trúc cơ bản khác với **MongoDB**. Thay vì cần có polling để thay đổi, lập trình viên có thể yêu cầu **RethinkDB** liên tục cập nhật các truy vấn kết quả theo thời gian thực. Bên cạnh đó, cũng có thể viết ứng dụng trên RethinkDB bằng cách sử dụng mô hình phản hồi truy vấn truyền thống và đăng kí nguồn dữ liệu thời gian thực, sau đó thêm chức năng thời gian thực vào ứng dụng.

Kiến trúc thời gian thực của **RethinkDB** có thể được so sánh với *oplog* của **MongoDB**, nhưng ở mức độ trừu tượng cao hơn nhiều. Nguồn cấp dữ liệu của **RethikDB** tích hợp liền mạch với công cụ tính toán truy vấn, cho phép thay đổi trực tiếp trên kết quả truy vấn chứ không chỉ là dữ liệu sao chép thô. Kiến trúc này sẽ giảm đáng kể thời gian và tài nguyên cần thiết để xây dựng các ứng dụng thời gian thực có thể mở rộng.

Bên cạnh đó, **RethinkDB** cũng có một số lợi thế khác so với **MongoDB:**

* Ngôn ngữ truy vấn nâng cao hỗ trợ kết hợp bảng, truy vấn con và tính toán phân tán song song hang loạt.
* Giao diện người dùng quản trị đơn giản, đẹp mắt, cho phép chia nhỏ và sao chép trong vài cú nhấp chuột, đồng thời cung cấp tài liệu trực tuyến cũng như đề xuất ngôn ngữ truy vấn.

1. **RETHINKDB KHÔNG PHẢI LUÔN LÀ LỰA CHỌN TỐT !**

**RethinkDB** sẽ không là lựa chọn tốt nếu bạn cần một hệ quản trị mang đầy đủ tính chất ACID hoặc cần thực thi các lược đồ quan hệ chặt chẽ. Trong trường hợp này, các hệ quản trị như MySQL, PostgreSQL hay Microsoft SQL Server sẽ phù hợp hơn.

Nếu như bạn có nhu cầu phân tích chuyên sâu về tính toán, tốt hơn hết bạn nên dung hệ thống như Hadoop hoặc Vertica.

Trong một số trường hợp, **RethinkDB** sẽ trade-off khả năng ghi để có lợi nhất cho tính nhất quán của dữ liệu. Do đó, nếu khả năng ghi là quan trọng và bạn không ngại giải quyết các xung đột, bạn có thể sử dụng hệ thống kiểu Dynamo như Riak.

1. **CÁC VẤN ĐỀ THỰC TIỄN XOAY QUANH RETHINKDB**
2. **CÁC NGÔN NGỮ CÓ THỂ SỬ DỤNG CÙNG VỚI RETHINKDB**

**RethinkDB** được nhà phát triển cung cấp các driver kết nối cho hang loạt ngôn ngữ như Ruby, Python, JavaScript/NodeJS. Ngoài ra, cũng có một số driver ngoài luồng hỗ tor75 cho Go, C#/.NET và PHP.

1. **CÁC YÊU CẦU LIÊN QUAN ĐẾN HỆ THỐNG ĐỂ SỬ DỤNG RETHINKDB**

Máy chủ **RethinkDB** được viết bằng ngôn ngữ C++ và chạy trên các hệ thống Linux 32 bit và 64 bit, cũng như OS X 10.7 trở lên. Trình điều khiển Client có thể chạy trên các nền tảng này.

Khuyến nghị: máy chủ RethinkDB nên có ít nhất 2GB RAM, ngoài ra không có yêu cầu nghiêm ngặt khác về phần cứng. RethinkDB có bộ nhớ đệm tùy chỉnh và có thể chạy trên các bộ nhớ thấp với lượng lớn dữ liệu trên đĩa, phiên bản Amazon EC2 … Cùng với đó, **RethinkDB** cũng có hỗ trợ chuyên biệt cho phần cứng cao cấp và hoạt động tốt trên nút bộ nhớ cao có nhiều lõi, bộ nhớ ở trạng thái rắn hoặc phần cứng mạng có thông lượng cao.

1. **CÁC TRANSACTIONS CỦA RETHINKDB CÓ TÍNH ATOMIC KHÔNG?**

Hầu hết các hoạt động ghi liên quan đến một tài liệu duy nhất trong RethinkDB sẽ được đảm bảo tính nguyên tử. Các transactions không xác định sẽ không thể cập nhật tài liệu theo kiểu nguyên tử (ví dụ như giá trị ngẫu nhiên, giá trị có được do kết quả truy vấn con…)

1. **TRƯỜNG HỢP DIRTY READ CÓ XẢY RA VỚI RETHINKDB KHÔNG?**

Các lần đọc của **RethinkDB** sẽ được chạy với option *read\_mode* **single** mặc định thường sẽ không thấy dữ liệu cũ, nhưn chúng có thể thấy các thay đổi từ các lần ghi đồng thời chưa được xác nhận an toàn trên đĩa. Điều này giống như mức cô lập ***READ UNCOMMITTED*** của SQL.

Để đảm bảo tính nhất quán của dữ liệu, bạn nên thay đổi *read\_mode* thành **majority**. Khi đó, **Dirty Read** sẽ không xảy ra, nhưng quá trình đọc sẽ chậm hơn.