# Tìm hiểu RabbitMQ - Phần 1

rabbitMQ 2 queue 4 message broker 2

## manhdung viết ngày 28/05/2015



**manhdung**
44 bài viết.
274 người follow





# Giới thiệu

RabbitMQ là một message broker ( message-oriented middleware) sử dụng giao thức AMQP - Advanced Message Queue Protocol (Đây là giao thức phổ biến, thực tế rabbitmq hỗ trợ nhiều giao thức). RabbitMQ được lập trình bằng ngôn ngữ Erlang. RabbitMQ cung cấp cho lập trình viên một phương tiện trung gian để giao tiếp giữa nhiều thành phần trong một hệ thống lớn ( Ví dụ openstack - Một công nghệ rất thú vị hi vọng một ngày nào đó tôi đủ sức để viết vài bài về chủ đề này ). RabbitMQ sẽ nhận message đến từ các thành phần khác nhau trong hệ thống, lưu trữ chúng an toàn trước khi đẩy đến đích.

Thực sự, với lập trình viên thì rabbitmq rất đáng giá. Nếu không có các hệ thống message broker như rabbitmq thì bất cứ lúc nào cần đẩy data giữa các thành phần trong hệ thống, lập trình viên cần một kết nối trực tiếp. Một hệ thống càng lớn. Số thành phần càng nhiều, mức độ trao đổi message giữa càng thành phần cũng vì thế tăng lên khiến việc lập trình trở nên phức tạp. Tôi từng đọc vài bài báo về lập trình thì thấy họ khuyến cáo các lập trình viên chỉ nên tập trung vào business logic của ứng dụng còn các công tác hậu trường thì nên được tái sử dụng các giải pháp đã có. Rabbitmq cũng là một giải pháp rất tốt trong các kiến trúc hệ thống lớn.

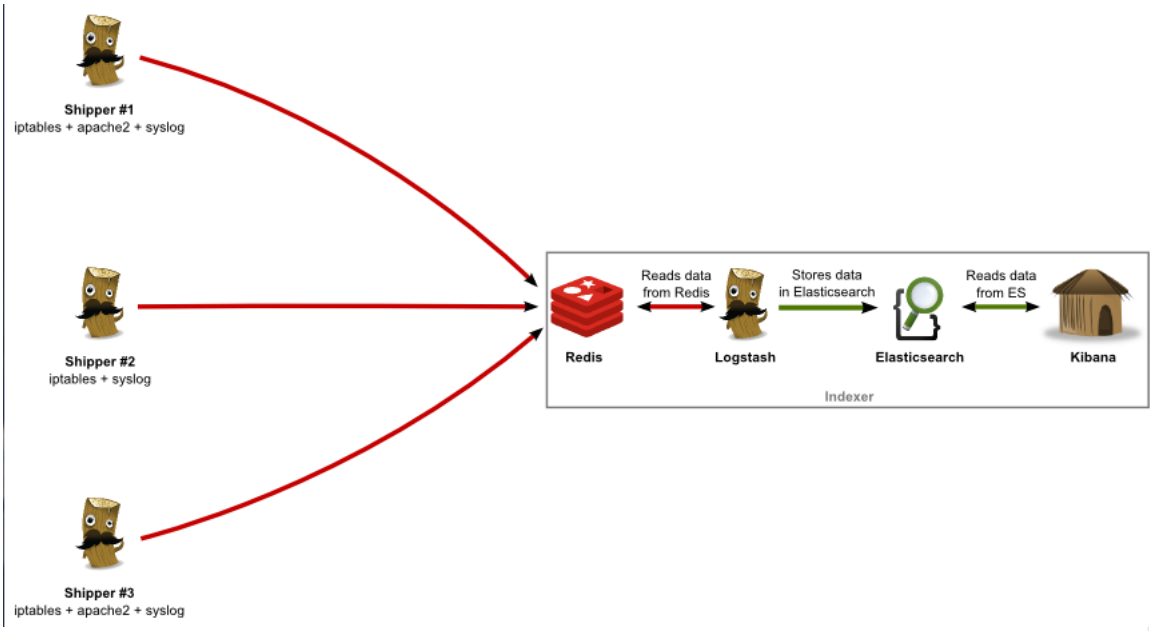
# Tại sao lại sử dụng RabbitMQ

Chúng ta thử xem các message broker như rabbitmq đem lại lợi ích gì trong việc thiết kế ứng dụng. Trong một hệ thống phân tán (distributed system), có rất nhiều thành phần. Nếu muốn các thành phần này giao tiếp được với nhau thì chúng phải biết nhau. Nhưng điều này gây rắc rối cho việc viết code. Một thành phần phải biết quá nhiều đâm ra rất khó maintain, debug. Giải pháp ở đây là thay vì các liên kết trực tiếp, khiến các thành phần phải biết nhau thì sử dụng một liên kết trung gian qua một message broker. Với sự tham gia của message broker thì producer sẽ không hề biết consumer. Nó chỉ việc gửi message đến các queue trong message broker. Consumer chỉ việc đăng ký nhận message từ các queue này.

Tất nhiên, có thể có một giải pháp là sử dụng database để lưu các message trong các temporary table. Tuy nhiên xét về hiệu năng thì không thể bằng message broker vì một số lý do: Tần xuất trao đổi message cao sẽ làm tăng load của database, giảm performance đáng kể. Trong môi trường multithread, database cần có cơ chế lock. Lock cũng làm giảm performance. Sử dụng message broker sẽ không có vấn đề này.

Vì producer nói chuyện với consumer trung gian qua message broker nên dù producer và consumer có khác biệt nhau về ngôn ngữ thì giao tiếp vẫn thành công. Dù viết bằng java, python, php hay ruby... thì chỉ cần thỏa mãn giao thức với message broker thì thông suốt hết. HIện nay, rabbitmq cũng đã cung cấp client library cho khá nhiều các ngôn ngữ rồi. Tính năng này cho phép tích hợp hệ thống linh hoạt.

Một đặc tính của rabbitmq là asynchronous. Producer không thể biết khi nào message đến được consumer hay khi nào message được consumer xử lý xong. Đối với producer, đẩy message đến message broker là xong việc. Consumer sẽ lấy message về khi nó muốn. Đặc tính này có thể được tận dụng để xây dựng các hệ thống lưu trữ và xử lý log. ELK stack - Elasticsearch Logstash Kibana là một ví dụ. Đây là hệ thống được sử dụng trong môi trường DevOps khá hiệu quả. Giải quyết bài toán chia sẻ log của ứng dụng cho dev. Log được đẩy vào message broker rồi qua logstash lưu trữ vào elastic để đánh index. Sau đó index được kibana (một web interface) sử dụng để thực hiện truy vấn và hiển thị kết quả

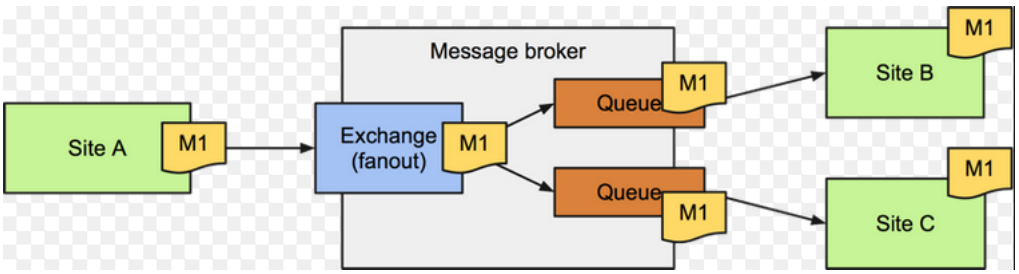



Bản thân redis cũng có thể đảm nhận vai trò của message broker ( Dù cho mục đích ban đầu của nó không phải vậy. Đây là tính năng mở rộng sau này của redis ) Mô hình trên bạn có thể hoàn toàn thay redis queue bằng rabbitmq.

Bên cạnh các lợi ích kể trên, rabbitmq còn có nhiều tính năng thú vị khác như:

- cluster: các bạn có thể gom nhiều rabbitmq instance vào một cluster. Một queue được đinh nghĩa trên một instance khi đó đều có thể truy xuất từ các instance còn lại. Có thể tận dụng để làm load balancing (tuy rằng có hạn chế, tôi sẽ nói sau)
- high availibilty: cho phép failover khi sử dụng mirror queue.
- reliability: có cơ chế ack để đảm bảo message được nhận bởi consumer đã được xử lý.

# Mô hình



Có thể hiểu message broker gần như bưu điện. Site A theo cách gọi của rabbitmq là producer (người gửi thông điệp). Site B và Site C theo cách gọi của rabbitmq là consumer (người nhận thông điệp). Producer connect đến message broker để đẩy message. Message sẽ đi qua message broker để đến được consumer. Cấu trúc của message broker chỉ gồm hai phần exchange và queue.

Exchange có nhiều loại. Trong hình vẽ trên exchange type là fanout. Lựa chọn các exchange type khác nhau sẽ dẫn đến khác đối xử khác nhau của message broker với thông điệp nhận được từ producer. Exchange được bind (liên kết) đến một số queue nhất định. Với exchange type là fanout, message sẽ được broadcast đến các queue được bind với exchange. Consumer sẽ connect đến message broker để lấy message từ các queue.

# Cài đặt và cấu hình cơ bản

Các bạn có thể tham khảo hướng dẫn trực tiếp từ trang chủ của rabbitmq:
https://www.rabbitmq.com/install-rpm.html

Vì rabbitmq được lập trình bằng erlang nên bạn cần cài đặt erlang trước. Tôi cài đặt erlang từ epel repo:

```
.org/pub/epel/6/i386/epel-release-6-8.noarch.rpm
m install --enablerepo=epel erlang
```

Sau đó, tôi cài tiếp rabbitmq:

```
tmq-server-3.5.2-1.noarch.rpm
tmq-signing-key-public.asc
m -ivh rabbitmq-server-3.5.2-1.noarch.rpm
```

VIệc cài đặt đến đây là xong. Bạn có thể start thử service của rabbitmq bằng cách:

```
service rabbitmq-server start
```

Cấu hình mẫu của rabbitmq đi kèm gói cài đặt nằm ở:
/usr/share/doc/rabbitmq-server-3.5.2/
Tôi copy về khu vực đặt file cấu hình của rabbitmq rồi đổi tên:

```
rabbitmq/
 /etc/rabbitmq/rabbitmq.config.example  /etc/rabbitmq/rabbitmq.config
```

Cấu hình của rabbitmq mặc định chạy khá ổn. Các bạn có thể sử dụng luôn mà không cần bận tâm tùy chỉnh cấu hình.

# Cài đặt management plugin

Rabbitmq có đi kèm một plugin cho phép quản trị hoạt động qua một web interface trông rất trực quan và thân thiện. Nhưng mặc định, plugin này không được enable. Để enable, bạn thực hiện lệnh sau:

```
rabbitmq-plugins enable rabbitmq_management
```

Tất cả các hoạt động quản trị qua web cũng có thể thực hiện qua một command line tool có tên là rabbitmqadmin. Tool này đi kèm trong management plugin. Bạn cũng chỉ có thể sử dụng nó sau khi đã enable management plugin.

Cách download rabbitmqadmin tool:

Tôi cài đặt rabbitmq trên một server có ip 192.168.3.252 nên địa chỉ download sẽ là:

```
http://192.168.3.252:15672/cli/rabbitmqadmin
```

Cách sử dụng tool này trong các tình huống cụ thể sẽ nằm trong các phần tới.

Trần Thành hơn 3 năm

Mình đang cần tìm hiểu cái này - bài nào của bạn cũng rất hay và chất lượng 👍

Hay
1

cpplover hơn 3 năm

> Tôi từng đọc vài bài báo về lập trình thì thấy họ khuyến cáo các lập trình viên chỉ nên tập trung vào business logic của ứng dụng còn các công tác hậu trường thì nên được tái sử dụng các giải pháp đã có. Rabbitmq cũng là một giải pháp rất tốt trong các kiến trúc hệ thống lớn.

Tớ thấy cái ý này rất đúng nhé.

Hay
1

Quăng hơn 3 năm

Bạn có thể nói thêm là các bài toán nào nên sử dụng một hệ thống Message_broker được không.

Hay
1

manhdung hơn 3 năm

@xluffy Tớ sẽ trả lời câu hỏi của bạn trong phần “Tại sao lại sử dung rabbitmq ?” (bổ sung thêm vào chính bài viết này) nhé

Hay
0

viettuan1807 hơn 3 năm

Trước đây mình cũng làm trên hệ thống FX có sử dụng ActiveMQ và ZeroMQ, về cơ bản sử sử dụng các hệ thống Message Broker rất tiện khi phải giao tiếp giữa nhiều thành phần của hệ thống sử dụng các ngôn ngữ lập trình khác nhau. Bạn nào thích tìm hiểu về các loại Message Broker thì có thể vào http://queues.io/ tham khảo.

Hay
1

Lợi Rệ hơn 3 năm

Mình thích bài giới thiệu này của bạn, dễ hiểu và trực quan. Mong bạn sẽ cung cấp thêm các lời khuyên về cách bảo trì và scale dịch vụ này trong các phần tới

Hay
0

tringq hơn 3 năm

cho mình hỏi nếu áp dùng gọi là Microservices Architecture thì RabbitMQ có thể đứng ở giữa tạo sự tương tác giữa thành phần với nhau không ? Mình nghĩ cơ bản về thật chất mọi thứ trao đổi với nhau là "message". Chúng ta đưa vào input ->output đầu là một "message" mà ta mong muốn.

Hay
0

manhdung hơn 3 năm

@tringq Được chứ bạn. Microservice architecture cũng là một dạng distributed system mà.

Bạn xem link này: http://www.infoq.com/articles/microservices-intro

Trong đó có giới thiệu hai cách giao tiếp chủ yếu trong microservice architecture: synchronous HTTP và asynchronous messaging. Cách thứ hai chính là sử dụng message broker.

Hay
1

Tú Phạm hơn 3 năm

Tiện nói về micro service thì mình thường setup rabbit mq qua docker

docker pull rabbitmq:3.5.3-management
docker run -d -e RABBITMQ_NODENAME=rabbitmq1 --name my-rabbitmq -p 5672:5672 -p 15672:15672 rabbitmq:3.5.3-management

Have fun :D

Hay
1

Minh Duc 1 năm

Mình đang xây dựng 1 hệ thống notification có sử dụng rabbitmq và đang gặp 1 số vấn đề khó khăn như sau.

1. hệ thống của mình cần gửi/nhận message đến các application the từng topic. Tính năng này tương đối đơn giản vì rabbitmq hầu như đã support hết.
2. đây mới là vấn đề: hệ thống cần lưu lại lịch sử (wirte log) của những app gửi nhận message. Ví dụ như khi 1 app đăng ký lắng nghe message (Consumer) thì cần lưu lại là app nào lắng nghe và thời điểm lắng nghe vào database. Hoặc khi 1 app nào đó gửi message (publiscer) thì cũng cần lưu lại thời điểm gửi message, nội dung message vào database. ==> cần có 1 server ngoài rabbitmq (mình gọi là server B) để lưu log vào database. Tuy nhiên việc giao tiếp giữa rabbitmq và con server B này sẽ làm như thế nào ?. Nếu coi server B là 1 Consumer lắng nghe toàn bộ message từ bất cứ app nào, thì làm thế nào để server B biết là đang có 1 app nào đó vừa đăng ký lắng nghe ?. Và kể cả nếu server B lấy được log, thì làm sao để giải quyết bài toán về performance ?

Tóm lại hiện tại mình chưa có giải pháp nào tốt cho bài toán này cả, nếu bạn thấy hứng thú có thể trao đổi với mình chứ ?

Hay
0

Minh Hải 1 năm

@MinhDuc11111 Khi có 1 application nào đó consume 1 queue thì sẽ có 1 cái đc sinh ra như comsumer tag, id comsumer. Có thể dựa vào đây là write log theo app. Hoặc cũng có thể message gửi lên đính kèm theo id để trace

Hay
0

Đăng nhập để bình luận :)

**manhdung**

44 bài viết. 274 người follow

Kipalog Follow

Cùng một tác giả

48 4

## Tìm hiểu phần cứng qua các thông tin server

hardware Server

Giả định bạn tiếp nhận một server mới toanh, bạn cần tìm một số thông tin về nó như loại CPU, loại main, loại memory, memory dùng của hãng nào... c...
manhdung viết 2 năm trước
48 4

43 12

## Tìm hiểu RabbitMQ - Phần 2

rabbitMQ message broker queue

Trong phần 1, tôi đã giới thiệu về sơ lược rabbitmq, vai trò của rabbitmq trong hệ thống phân tán và hướng dẫn cài đặt. Trong phần này, tôi sẽ trìn...
manhdung viết hơn 3 năm trước
43 12

41 7

## Giới thiệu MongoDB

mongodb

Giới thiệu MongoDB là một giải pháp nosql database. Data được lưu ở dạng các bson document. Hỗ trợ vertical scaling và horizontal scaling, dynamic...
manhdung viết hơn 3 năm trước
41 7

Bài viết liên quan

43 12

## Tìm hiểu RabbitMQ - Phần 2

rabbitMQ message broker queue

Trong phần 1, tôi đã giới thiệu về sơ lược rabbitmq, vai trò của rabbitmq trong hệ thống phân tán và hướng dẫn cài đặt. Trong phần này, tôi sẽ trìn...
manhdung viết hơn 3 năm trước
43 12