TRƯỜNG ĐẠI HỌC THỦY LỢI

KHOA CÔNG NGHỆ THÔNG TIN



Môn học: Chuyên đề 4

Đề tài: Xây dựng ứng dụng đơn giản với RabbitMQ

Giảng viên hướng dẫn: TS.Nguyễn Ngọc Doanh

Thành viên : **Nguyễn Văn Phong - 1551060967**

Trần Thanh Tùng - 1551060675

Nguyễn Văn Đức - 1551060640

Lê Hồng Sơn - 1551061067

Nguyễn Ngọc Sơn - 1551060777

Lớp: 57TH4

Hà Nội,03/2019

Lời Nói Đầu

Ngày nay, khoa học công nghệ đang phát triển vượt bậc trên thế giới, con người đã đẩy mạnh được nhiều tiềm năng ứng dụng của ngành công nghệ thông tin trong thực tế cuộc sống. Cùng với đó là sự cải thiện nâng cấp các thiết bị phần cứng cũng như cơ sở hạ tầng để đáp ứng phù hợp với nhu cầu sử dụng của con người. Nhưng với nhu cầu ngày một tăng trong việc sử dụng dữ liệu, công nghệ thì việc phát triển đó chưa đủ nên con người ta cần phải có một phương pháp để giải quyết các bài toán mang tính phức tạp hơn. Các bài toán yêu cầu sự phân luồng các quá trình để đưa vào hàng đợi. Chính vì vậy **RabbitMQ** được sinh ra để đáp ứng được điều đó. **RabbitMQ** là một phần mềm quản-lý-hàng-đợi message, thường được gọi là môi-giới-message hay trình-quản-lý-message. Nói đơn giản, đây là phần mềm định nghĩa hàng đợi một ứng dụng khác có thể kết nối đến để bỏ message vào và gửi message dựa trên nó. Chính vì thế RabbitMQ đã trở thành một hệ thống được dung nhiều cho các công ty hiện nay. Bài viết sau đây sẽ giới thiệu cho ta biết về RabbitMQ cũng như ứng dụng của nó trong thực tiễn cuộc sống.

1. **RabbitMQ**

- RabbitMQ là một hệ thống xếp hàng tin nhắn phân tán. Phân phối vì nó thường được chạy dưới dạng một cụm các nút trong đó hàng đợi được trải rộng trên các nút và được nhân rộng tùy ý cho khả năng chịu lỗi và tính sẵn sàng cao. Nó thực sự thực hiện AMQP 0.9.1 và cung cấp các giao thức khác như STOMP, MQTT và HTTP thông qua các trình cắm.

- RabbitMQ có cả một tác phẩm kinh điển và tiểu thuyết về nhắn tin. Cổ điển theo nghĩa là nó được định hướng xung quanh hàng đợi tin nhắn và tiểu thuyết trong khả năng định tuyến rất linh hoạt của nó. Chính khả năng định tuyến này là tính năng sát thủ của nó. Xây dựng một hệ thống nhắn tin phân tán nhanh, có thể mở rộng, đáng tin cậy là một thành tựu, nhưng chức năng định tuyến tin nhắn là điều làm cho nó thực sự nổi bật giữa vô số công nghệ nhắn tin ngoài kia.

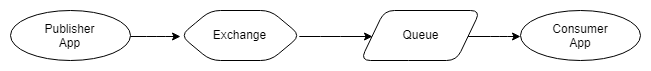
- Rabbit: Trao đổi và xếp hàng

*a. Tổng quan đơn giản:*

* Các nhà xuất bản gửi tin nhắn để trao đổi
* Trao đổi tin nhắn định tuyến đến hàng đợi và trao đổi khác
* RabbitMQ gửi xác nhận đến các nhà xuất bản khi nhận được tin nhắn
* Người tiêu dùng duy trì các kết nối TCP liên tục với RabbitMQ và khai báo hàng đợi họ tiêu thụ
* RabbitMQ đẩy tin nhắn đến người tiêu dùng
* Người tiêu dùng gửi lời cảm ơn về thành công / thất bại
* Tin nhắn được xóa khỏi hàng đợi một khi được tiêu thụ thành công

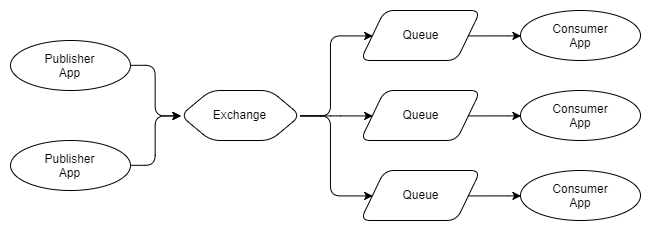
Ẩn trong danh sách đó là một số lượng lớn các quyết định mà các nhà phát triển và quản trị viên nên thực hiện để có được sự đảm bảo phân phối mà họ muốn, đặc tính hiệu suất, v.v., tất cả chúng ta sẽ đề cập trong các phần sau của loạt bài này.

Chúng ta hãy xem một nhà xuất bản, trao đổi, xếp hàng và người tiêu dùng:



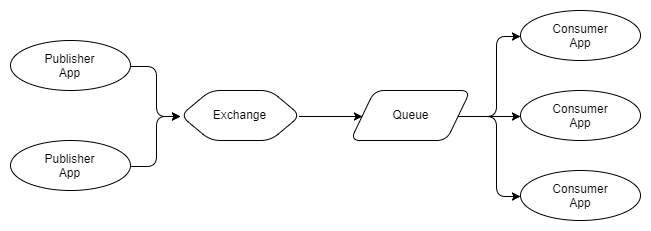
Hình 1 - Nhà xuất bản đơn và người tiêu dùng duy nhất

Điều gì nếu bạn có nhiều nhà xuất bản của cùng một tin nhắn? Ngoài ra, nếu chúng ta có nhiều người tiêu dùng mà mỗi người muốn tiêu thụ mọi thông điệp thì sao?



Hình 2 - Nhiều nhà xuất bản, nhiều người tiêu dùng độc lập

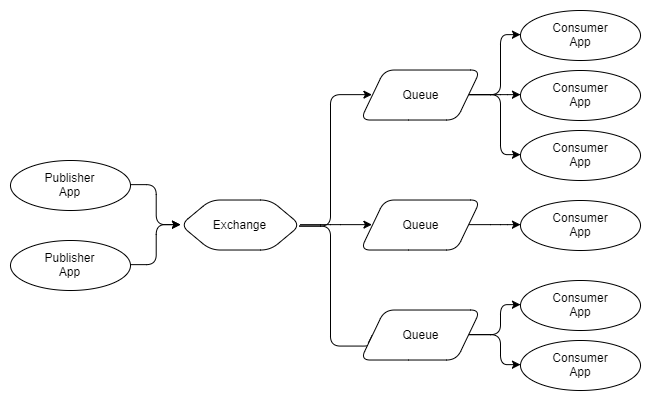
- Như bạn có thể thấy, các nhà xuất bản gửi tin nhắn của họ đến cùng một trao đổi, định tuyến mỗi tin nhắn đến ba hàng đợi, mỗi hàng có một người tiêu dùng. Với RabbitMQ, hàng đợi cho phép người tiêu dùng khác nhau tiêu thụ từng tin nhắn. Tương phản với sơ đồ dưới đây:



Hình 3 - Nhiều nhà xuất bản, một hàng đợi với nhiều người tiêu dùng cạnh tranh

-Trong hình 3, chúng ta có ba người tiêu dùng tất cả tiêu thụ từ một hàng đợi. Đây là những người tiêu dùng cạnh tranh, nghĩa là họ cạnh tranh để tiêu thụ các thông điệp của một hàng đợi. Người ta sẽ mong đợi rằng trung bình, mỗi người tiêu dùng sẽ tiêu thụ một phần ba số tin nhắn của hàng đợi này. Chúng tôi sử dụng người tiêu dùng cạnh tranh để mở rộng quy mô xử lý tin nhắn của chúng tôi và với RabbitMQ rất đơn giản, chỉ cần thêm hoặc xóa người tiêu dùng theo yêu cầu. Cho dù bạn có bao nhiêu người tiêu dùng cạnh tranh, RabbitMQ sẽ đảm bảo rằng các tin nhắn được gửi đến chỉ một người tiêu dùng.

Chúng ta có thể kết hợp hình 2 và 3 để có nhiều bộ khách hàng cạnh tranh trong đó mỗi bộ tiêu thụ mọi thông điệp.



Hình 4 - Nhiều nhà xuất bản, nhiều hàng đợi với người tiêu dùng cạnh tranh

Mũi tên giữa trao đổi và hàng đợi được gọi là các ràng buộc và chúng ta sẽ xem xét kỹ hơn những thứ trong Phần 2 của loạt bài này.

### *b. Đảm bảo*

RabbitMQ cung cấp “nhiều nhất một lần giao hàng” và *“* ít nhất một lần giao hàng” nhưng không đảm bảo “giao hàng chính xác một lần”.

Tin nhắn được **gửi** theo thứ tự đến hàng đợi (đó là định nghĩa của hàng đợi sau tất cả). Điều này không đảm bảo hoàn thành việc xử lý tin nhắn khớp với chính xác đơn hàng đó khi bạn có khách hàng cạnh tranh. Đây không phải là lỗi của RabbitMQ mà là một thực tế cơ bản của việc xử lý song song một tập hợp các thư được đặt hàng. Vấn đề này có thể được giải quyết bằng cách sử dụng Trao đổi băm nhất quán như bạn sẽ thấy trong phần tiếp theo về các mẫu và cấu trúc liên kết.

*c. Tiêu chuẩn và tiêu dùng*

RabbitMQ đẩy tin nhắn đến người tiêu dùng trong một luồng. Có một API kéo nhưng nó có hiệu suất khủng khiếp vì mỗi thông báo yêu cầu một chuyến đi khứ hồi yêu cầu / phản hồi.

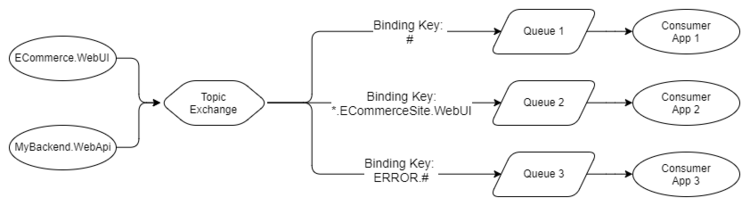
Các hệ thống dựa trên đẩy có thể áp đảo người tiêu dùng nếu tin nhắn đến hàng đợi nhanh hơn người tiêu dùng có thể xử lý chúng. Vì vậy, để tránh điều này, mỗi người tiêu dùng có thể định cấu hình giới hạn tìm nạp trước (còn được gọi là giới hạn QoS). Về cơ bản, đây là số lượng tin nhắn chưa được xác nhận mà người tiêu dùng có thể có bất cứ lúc nào. Điều này hoạt động như một công tắc cắt an toàn khi người tiêu dùng bắt đầu tụt lại phía sau.

Tại sao đẩy và không kéo? Trước hết nó là tuyệt vời cho độ trễ thấp. Thứ hai, lý tưởng khi chúng tôi có những khách hàng cạnh tranh của một hàng đợi duy nhất, chúng tôi muốn phân phối tải đồng đều giữa họ. Nếu mỗi người tiêu dùng kéo tin nhắn thì tùy thuộc vào số lượng họ kéo phân phối công việc có thể trở nên không đồng đều. Việc phân phối tin nhắn càng không đồng đều thì độ trễ càng nhiều và mất càng nhiều thứ tự tin nhắn trong thời gian xử lý. Vì lý do đó, API kéo của RabbitMQ chỉ cho phép kéo một tin nhắn tại một thời điểm, nhưng điều đó ảnh hưởng nghiêm trọng đến hiệu suất. Những yếu tố này làm cho RabbitMQ nghiêng về một cơ chế đẩy. Đây là một trong những hạn chế mở rộng của RabbitMQ. Nó được cải thiện bằng cách có thể nhóm các xác nhận với nhau.

*d. Định tuyến*

Trao đổi về cơ bản là bộ định tuyến của tin nhắn đến hàng đợi và / hoặc trao đổi khác. Để một tin nhắn đi từ một trao đổi đến một hàng đợi hoặc trao đổi khác, một ràng buộc là cần thiết. Trao đổi khác nhau đòi hỏi ràng buộc khác nhau. Có bốn loại trao đổi và ràng buộc liên quan:

* **Fanout** : Các tuyến đường đến tất cả các hàng đợi và trao đổi có ràng buộc với trao đổi. Các mô hình phụ pub tiêu chuẩn.
* **Trực tiếp**: Định tuyến thư dựa trên **Khóa định tuyến** mà thư mang theo, do nhà xuất bản đặt. Khóa định tuyến là một chuỗi ngắn. Trao đổi trực tiếp tin nhắn định tuyến đến hàng đợi / trao đổi có **Khóa Binding** khớp chính xác với khóa định tuyến.
* **Chủ đề**: Định tuyến tin nhắn dựa trên khóa định tuyến, nhưng cho phép khớp ký tự đại diện.
* **Tiêu đề**: RabbitMQ cho phép các tiêu đề tùy chỉnh được thêm vào tin nhắn. Tiêu đề trao đổi thông điệp định tuyến theo các giá trị tiêu đề. Mỗi ràng buộc bao gồm các giá trị tiêu đề phù hợp chính xác. Nhiều giá trị có thể được thêm vào một ràng buộc với bất kỳ hoặc tất cả các giá trị cần thiết để khớp.
* **Băm nhất quán**: Đây là một trao đổi băm hoặc khóa định tuyến hoặc tiêu đề thư và chỉ định tuyến đến một hàng đợi. Điều này hữu ích khi bạn cần xử lý đảm bảo đơn hàng với người tiêu dùng được thu nhỏ.



Hình 5. Ví dụ trao đổi chủ đề

Trên đây là một ví dụ về trao đổi Chủ đề. Các nhà xuất bản xuất bản nhật ký lỗi với định dạng Khóa định tuyến của LEVEL.AppName .

* Hàng đợi 1 sẽ nhận được tất cả các tin nhắn vì nó sử dụng ký tự đại diện # nhiều từ.
* Hàng đợi 2 sẽ nhận được bất kỳ cấp độ nhật ký nào của ứng dụng EC Commerce.WebUI. Nó sử dụng ký tự đại diện \* từ đơn bao gồm cấp độ nhật ký.
* Hàng đợi 3 sẽ thấy tất cả các thông báo cấp độ LRI từ bất kỳ ứng dụng nào. Nó sử dụng ký tự đại diện # từ để bao gồm tất cả các ứng dụng.

Với bốn cách định tuyến tin nhắn và cho phép các trao đổi định tuyến đến các trao đổi khác, RabbitMQ cung cấp một tập hợp các mẫu tin nhắn mạnh mẽ và linh hoạt. Tiếp theo chúng ta sẽ chạm vào trao đổi thư chết, trao đổi phù du và hàng đợi, và bạn sẽ bắt đầu thấy sức mạnh của RabbitMQ.

*e. Trao đổi thư chết*

Chúng tôi có thể định cấu hình hàng đợi để gửi tin nhắn đến một trao đổi theo các điều kiện sau:

* Hàng đợi vượt quá số lượng tin nhắn được cấu hình.
* Hàng đợi vượt quá số byte được cấu hình.
* Thời gian nhắn tin để sống (TTL) đã hết hạn. Nhà xuất bản có thể đặt thời gian tồn tại của tin nhắn và hàng đợi cũng có thể có tin nhắn TTL. Mà bao giờ là ngắn hơn áp dụng.
* Chúng tôi tạo ra một hàng đợi có ràng buộc với trao đổi thư chết và những thông điệp này được lưu trữ ở đó cho đến khi hành động được thực hiện. Trong một bài đăng riêng biệt, tôi đã mô tả cấu trúc liên kết mà tôi đã thực hiện trong đó tất cả các tin nhắn thư chết đến một trung tâm thanh toán bù trừ trung tâm nơi nhóm hỗ trợ có thể quyết định những hành động cần thực hiện.
* Giống như nhiều chức năng của RabbitMQ, trao đổi thư chết cho các mẫu bổ sung không được xem xét ban đầu. Chúng ta có thể sử dụng tin nhắn TTL và trao đổi thư chết để thực hiện hàng đợi trễ và thử lại hàng đợi bao gồm cả số lùi. Xem bài viết trước của tôi về điều này.

*f. Trao đổi phù du và hàng đợi*

Trao đổi và hàng đợi có thể được tạo động và đưa ra các đặc điểm tự động xóa. Sau một khoảng thời gian nhất định chúng có thể tự hủy. Điều này cho phép các mẫu như hàng đợi trả lời ephermal cho RPC dựa trên thông báo.

*g. Bổ sung*

Trình cắm đầu tiên bạn muốn cài đặt là Trình cắm quản lý cung cấp máy chủ HTTP, với giao diện người dùng web và API REST. Nó thực sự dễ cài đặt và cung cấp cho bạn một giao diện người dùng dễ sử dụng để giúp bạn khởi động và chạy. Việc triển khai kịch bản thông qua API REST cũng thực sự dễ dàng.

Một số plugin khác bao gồm:

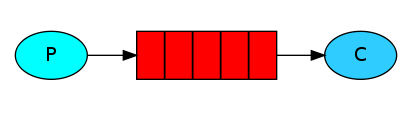
* Trao đổi băm nhất quán, trao đổi Shending và nhiều hơn nữa
* Các giao thức như STOMP và MQTT
* Móc web
* Loại trao đổi thêm
* Tích hợp SMTP

Có rất nhiều điều nữa cho RabbitMQ nhưng đó là một mồi tốt và cung cấp cho bạn ý tưởng về những gì RabbitMQ có thể làm. Bây giờ chúng ta sẽ xem xét Kafka, một cách tiếp cận hoàn toàn khác với nhắn tin, và cũng có các tính năng tuyệt vời.

Cài đặt RabbitMQ:

Trong phần hướng dẫn này, chúng tôi sẽ viết hai chương trình bằng Java; một nhà sản xuất gửi một tin nhắn và một người tiêu dùng nhận tin nhắn và in chúng ra. Chúng tôi sẽ giới thiệu một số chi tiết trong API Java, tập trung vào điều rất đơn giản này chỉ để bắt đầu. Đó là một "Hello World" của tin nhắn.

Trong sơ đồ bên dưới, "P" là nhà sản xuất của chúng tôi và "C" là người tiêu dùng của chúng tôi. Hộp ở giữa là một hàng đợi - một bộ đệm thông báo mà RabbitMQ giữ thay cho người tiêu dùng.



#### Thư viện máy khách Java

RabbitMQ nói nhiều giao thức. Hướng dẫn này sử dụng AMQP 0-9-1, đây là một giao thức mở, mục đích chung để nhắn tin. Có một số khách hàng cho RabbitMQ bằng nhiều ngôn ngữ khác nhau . Chúng tôi sẽ sử dụng máy khách Java được cung cấp bởi RabbitMQ.

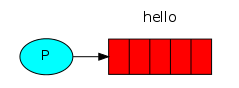
Tải xuống thư viện máy khách và các phụ thuộc của nó ( API SLF4J và SLF4J Simple ). Sao chép các tệp đó trong thư mục làm việc của bạn, cùng với các tệp Java hướng dẫn.

Xin lưu ý SLF4J Simple là đủ cho hướng dẫn nhưng bạn nên sử dụng thư viện ghi nhật ký đầy đủ như Logback trong sản xuất.

(Máy khách Java RabbitMQ cũng nằm trong kho lưu trữ Maven trung tâm, với com.rabbitmq của nhóm id và artqactId amqp-client.)

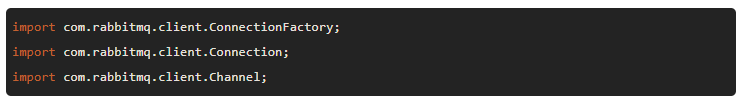
Bây giờ chúng ta có máy khách Java và các phụ thuộc của nó, chúng ta có thể viết một số mã.

Send



Chúng tôi sẽ gọi nhà xuất bản tin nhắn của chúng tôi (người gửi) Send và người tiêu dùng tin nhắn của chúng tôi (người nhận) Recv . Nhà xuất bản sẽ kết nối với RabbitMQ, gửi một tin nhắn, sau đó thoát.

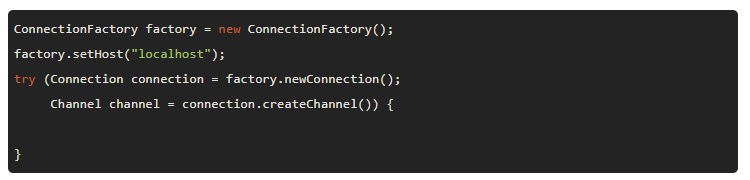
Trong Send.java , chúng tôi cần một số lớp được nhập:



Thiết lập lớp và đặt tên cho hàng đợi:



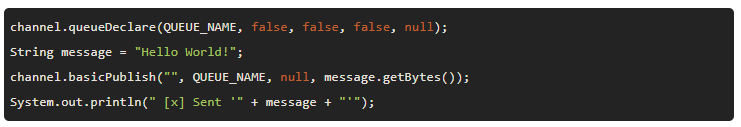
Sau đó chúng ta có thể tạo kết nối đến máy chủ:



Kết nối trừu tượng hóa kết nối ổ cắm, và đảm nhiệm việc đàm phán và xác thực phiên bản giao thức, v.v. Ở đây chúng tôi kết nối với một nhà môi giới trên máy cục bộ - do đó là localhost. Nếu chúng tôi muốn kết nối với một nhà môi giới trên một máy khác, chúng tôi chỉ cần chỉ định tên hoặc địa chỉ IP của nó ở đây.

Tiếp theo, chúng tôi tạo một kênh, nơi chứa hầu hết các API để hoàn thành công việc. Lưu ý rằng chúng tôi có thể sử dụng câu lệnh try-with-resource vì cả Connection và Channel triển khai java.io.Closizable. Bằng cách này, chúng tôi không cần phải đóng chúng rõ ràng trong mã của chúng tôi.

Để gửi, chúng tôi phải khai báo một hàng đợi để chúng tôi gửi đến; sau đó chúng ta có thể xuất bản một thông báo tới hàng đợi, tất cả những điều này trong câu lệnh try-with-resource:



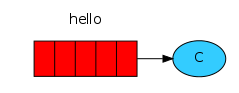
Khai báo một hàng đợi là idempotent - nó sẽ chỉ được tạo nếu nó chưa tồn tại. Nội dung tin nhắn là một mảng byte, vì vậy bạn có thể mã hóa bất cứ thứ gì bạn thích ở đó.

#### Gửi không hoạt động!

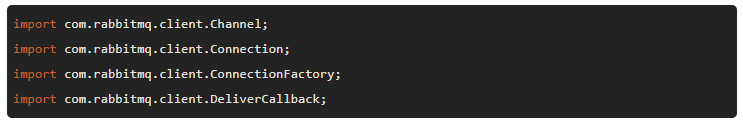
Nếu đây là lần đầu tiên bạn sử dụng RabbitMQ và bạn không thấy thông báo "Đã gửi" thì bạn có thể phải gãi đầu tự hỏi điều gì có thể sai. Có thể nhà môi giới đã được bắt đầu mà không có đủ dung lượng đĩa trống (theo mặc định, nó cần ít nhất 200 MB miễn phí) và do đó từ chối chấp nhận tin nhắn. Kiểm tra logfile môi giới để xác nhận và giảm giới hạn nếu cần thiết. Các tài liệu hướng dẫn tập tin cấu hình sẽ cho bạn thấy làm thế nào để thiết lập disk\_free\_limit.

Receiving

Đó là nó cho nhà xuất bản của chúng tôi. Người tiêu dùng của chúng tôi nghe tin nhắn từ RabbitMQ, vì vậy, không giống như nhà xuất bản phát hành một tin nhắn, chúng tôi sẽ tiếp tục chạy để nghe tin nhắn và in chúng ra.

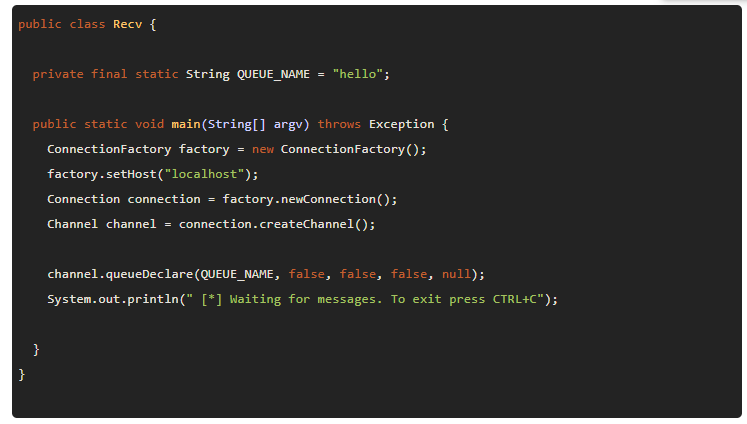


Mã (trong Recv.java ) có các lần nhập gần giống như Send :



DefaultConsumer bổ sung là một lớp triển khai giao diện Consumer, chúng tôi sẽ sử dụng để đệm các tin nhắn được máy chủ đẩy đến cho chúng tôi.

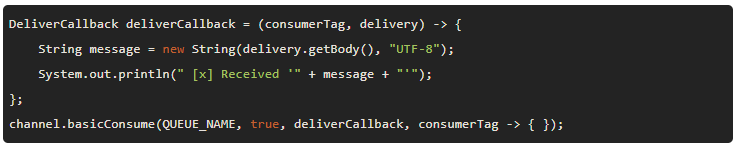
Thiết lập giống như nhà xuất bản; chúng tôi mở một kết nối và một kênh, và tuyên bố hàng đợi mà chúng tôi sẽ tiêu thụ. Lưu ý điều này khớp với hàng đợi Send xuất bản tới.



Lưu ý rằng chúng tôi cũng tuyên bố hàng đợi ở đây. Vì chúng tôi có thể bắt đầu người tiêu dùng trước nhà xuất bản, chúng tôi muốn đảm bảo hàng đợi tồn tại trước khi chúng tôi cố gắng tiêu thụ thư từ nó.

Tại sao chúng ta không sử dụng câu lệnh thử với tài nguyên để tự động đóng kênh và kết nối? Bằng cách làm như vậy, chúng tôi chỉ đơn giản sẽ làm cho chương trình tiếp tục, đóng mọi thứ và thoát! Điều này sẽ gây khó xử vì chúng tôi muốn quá trình này tồn tại trong khi người tiêu dùng đang lắng nghe không đồng bộ để có tin nhắn đến.

Chúng tôi sắp nói với máy chủ gửi cho chúng tôi các tin nhắn từ hàng đợi. Vì nó sẽ đẩy chúng ta các tin nhắn không đồng bộ, chúng tôi cung cấp một cuộc gọi lại dưới dạng một đối tượng sẽ đệm các tin nhắn cho đến khi chúng tôi sẵn sàng sử dụng chúng. Đó là những gì một lớp con DeliveryCallback làm.



### Putting it all together

Bạn có thể biên dịch cả hai thứ này chỉ bằng ứng dụng java RabbitMQ trên đường dẫn lớp:

C:\Users\Admin\Desktop\12.PNG

Để chạy chúng, bạn sẽ cần rabbitmq-client.jar và các phụ thuộc của nó vào đường dẫn lớp. Trong một thiết bị đầu cuối, chạy người tiêu dùng (người nhận):

C:\Users\Admin\Desktop\13.PNG

sau đó, chạy nhà xuất bản (người gửi):

C:\Users\Admin\Desktop\14.PNG

Trên Windows, sử dụng dấu chấm phẩy thay vì dấu hai chấm để phân tách các mục trong đường dẫn lớp.

Người tiêu dùng sẽ in thông điệp mà họ nhận được từ nhà xuất bản thông qua RabbitMQ. Người tiêu dùng sẽ tiếp tục chạy, chờ tin nhắn (Sử dụng Ctrl-C để dừng nó), vì vậy hãy thử chạy nhà xuất bản từ thiết bị đầu cuối khác.

#### Danh sách hàng đợi

Bạn có thể muốn xem hàng đợi RabbitMQ có gì và có bao nhiêu tin nhắn trong đó. Bạn có thể làm điều đó (với tư cách là người dùng đặc quyền) bằng công cụ rabbitmqctl :

C:\Users\Admin\Desktop\15.PNG

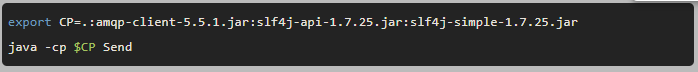
Trên Windows, bỏ qua sudo:

C:\Users\Admin\Desktop\16.PNG

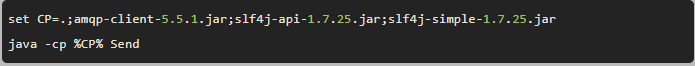
Thời gian để chuyển sang phần 2 và xây dựng một hàng đợi công việc đơn giản .

Hint

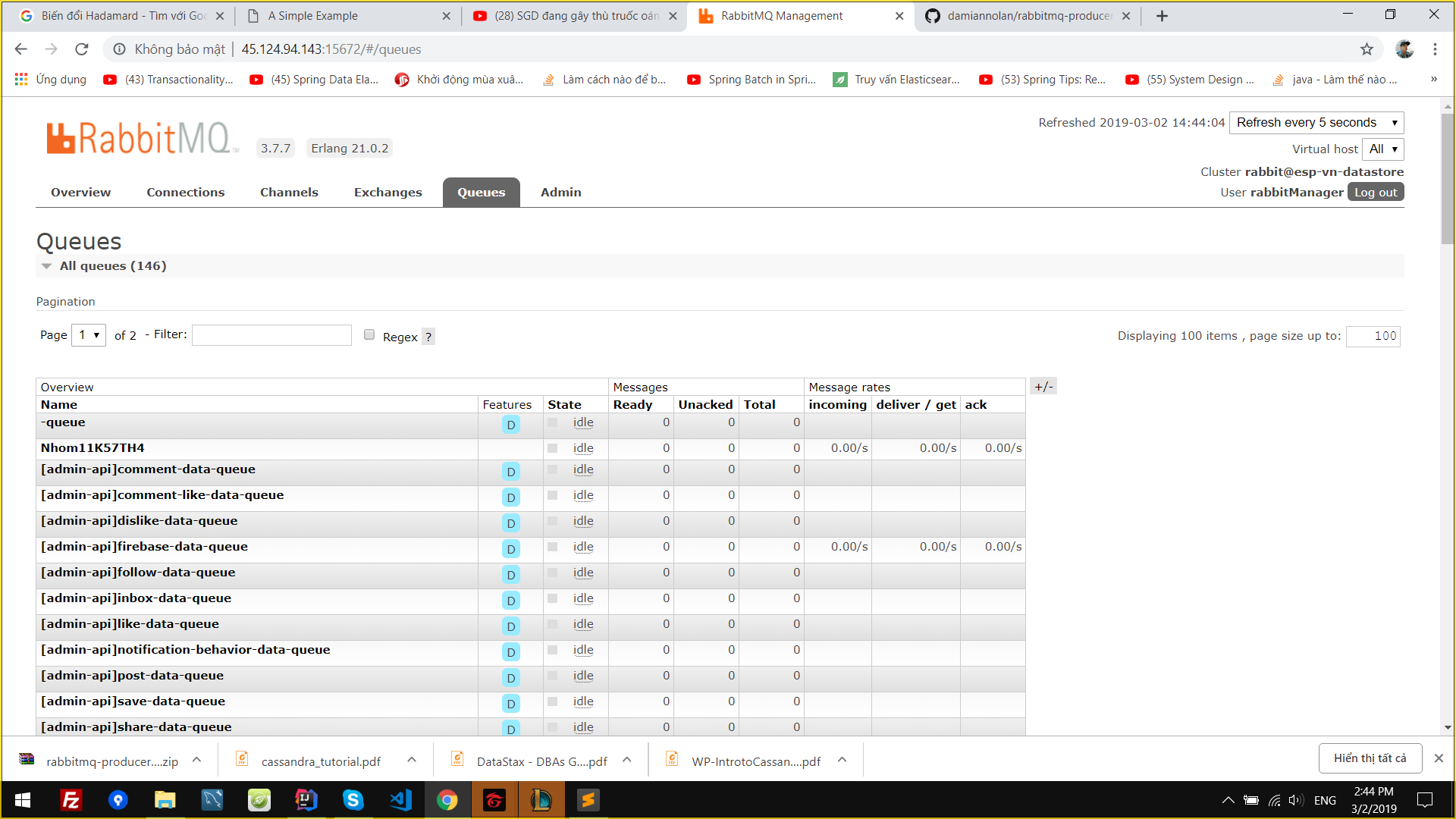
Để lưu kiểu gõ, bạn có thể đặt biến môi trường cho đường dẫn lớp, vd



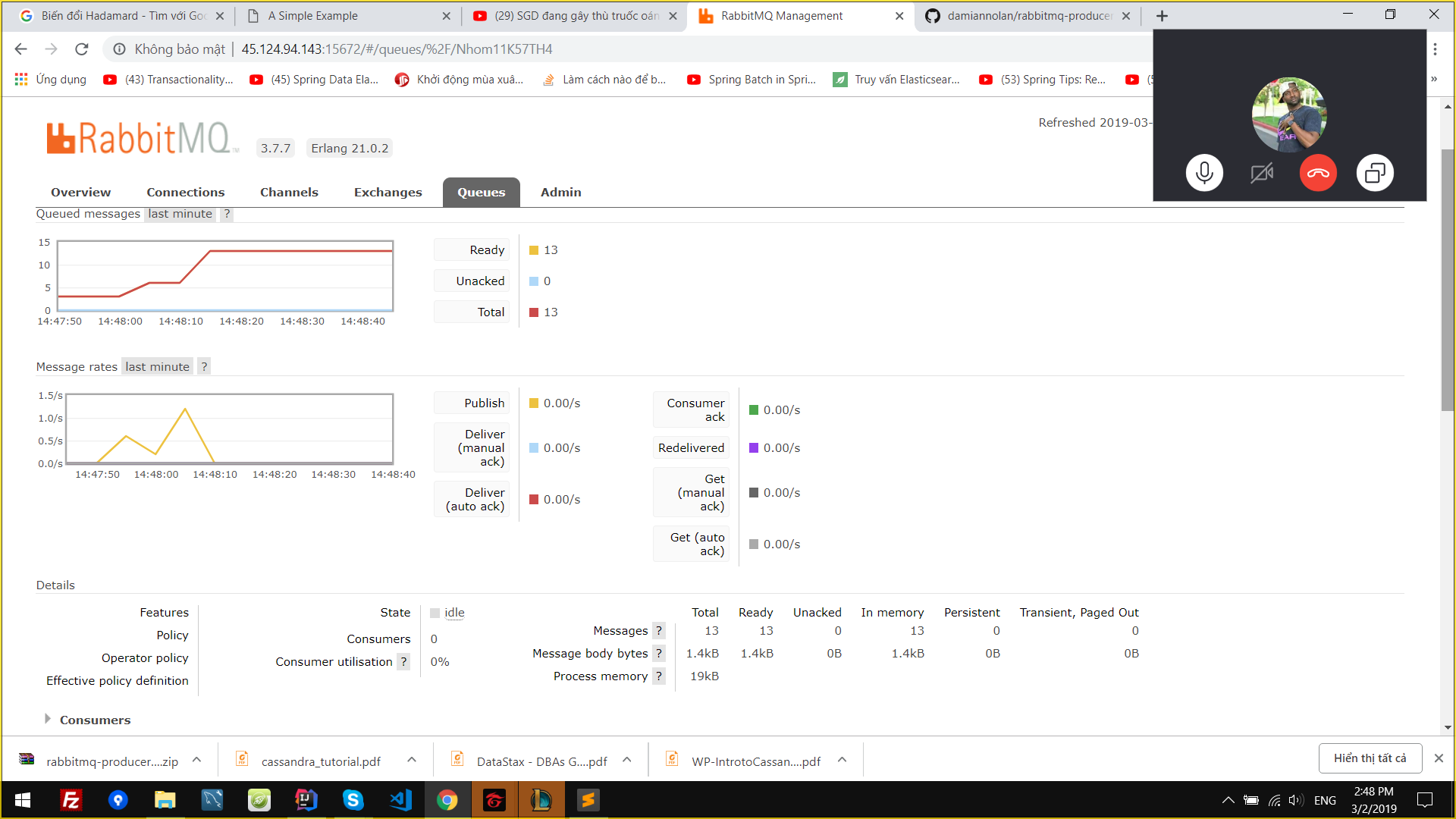
Hoặc trên Windows:



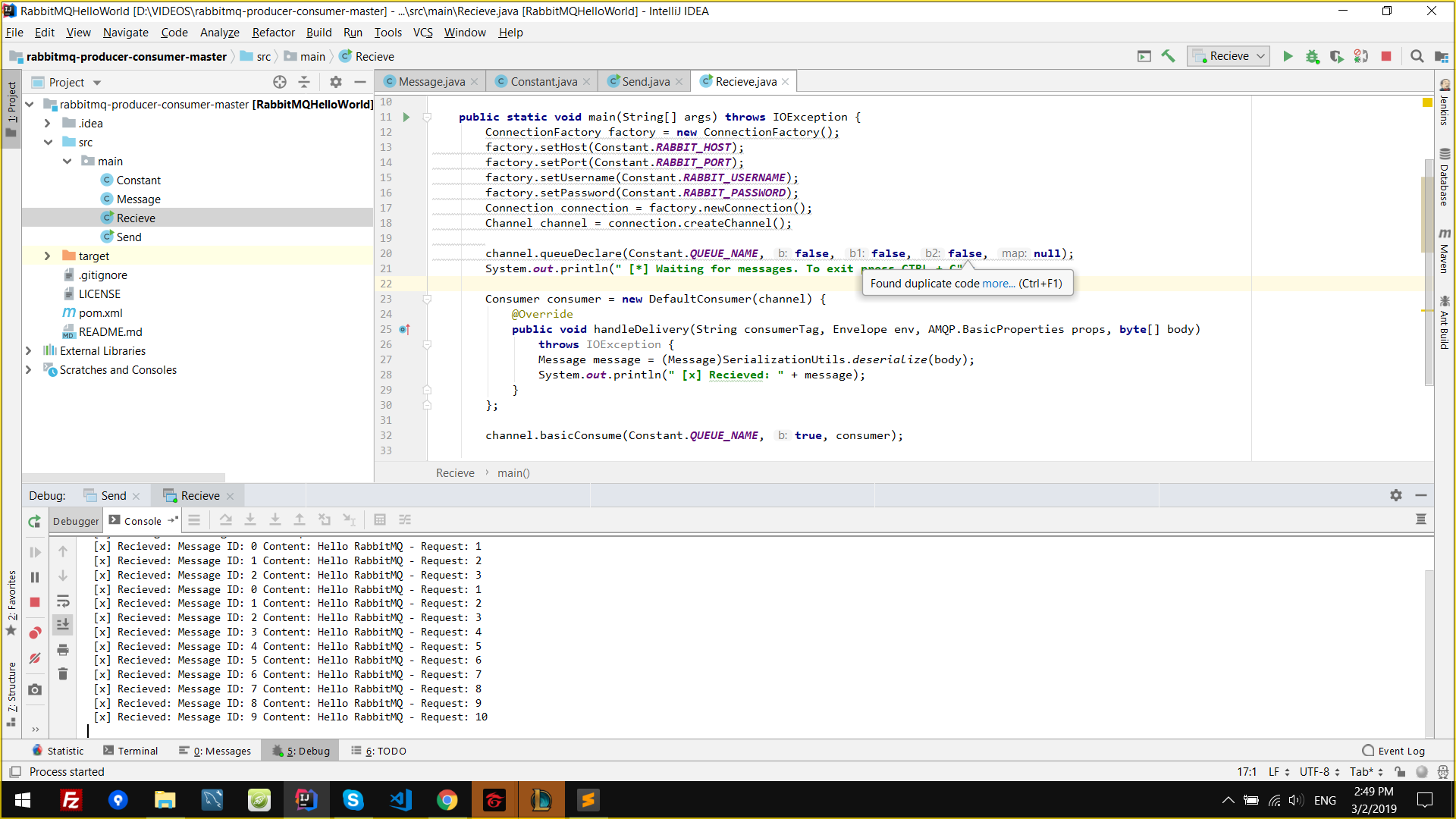
Sử dụng demo chương trình gửi nhận tin nhắn:

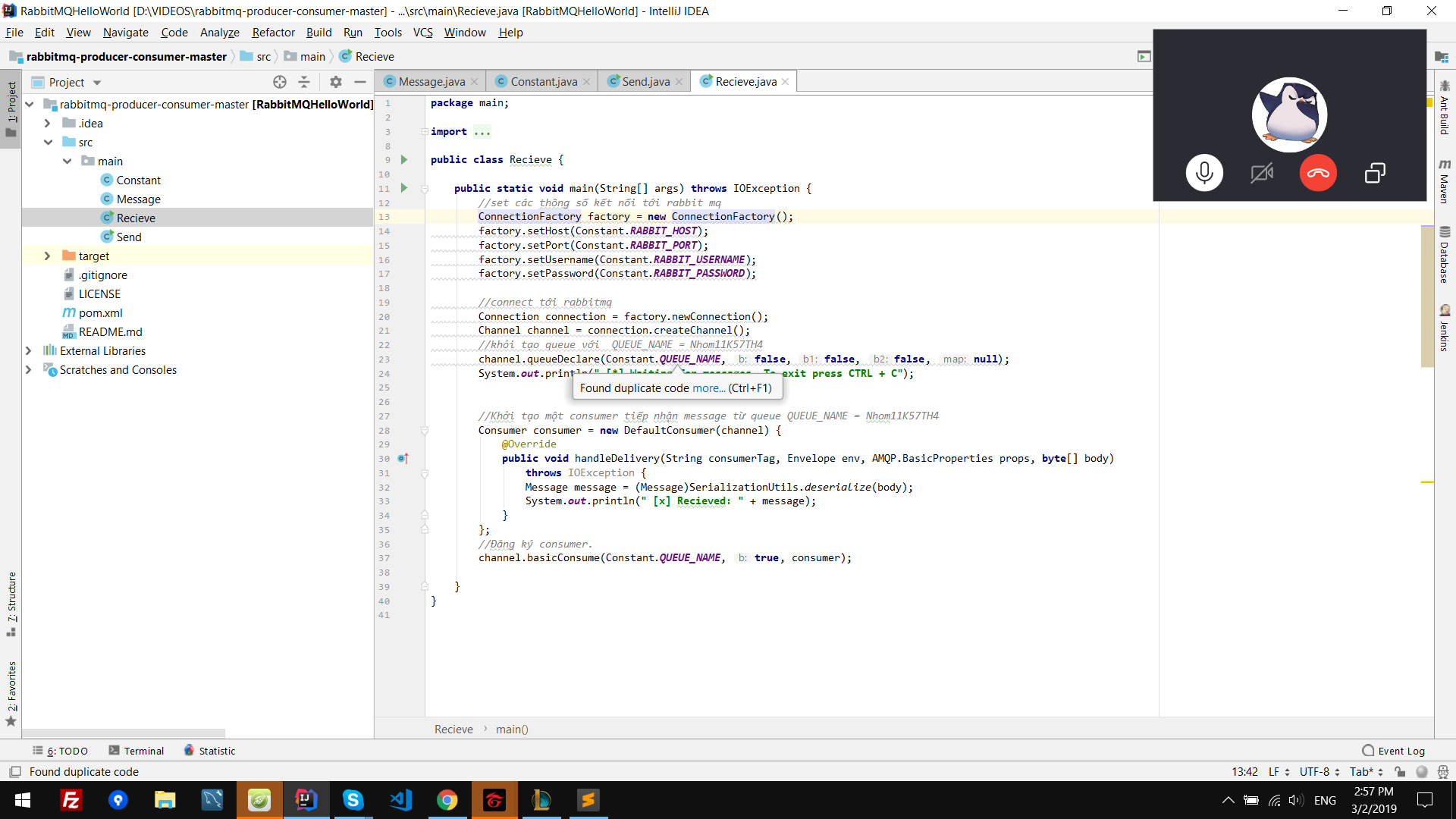


Hàng đợi tin nhắn đang chờ được gửi:

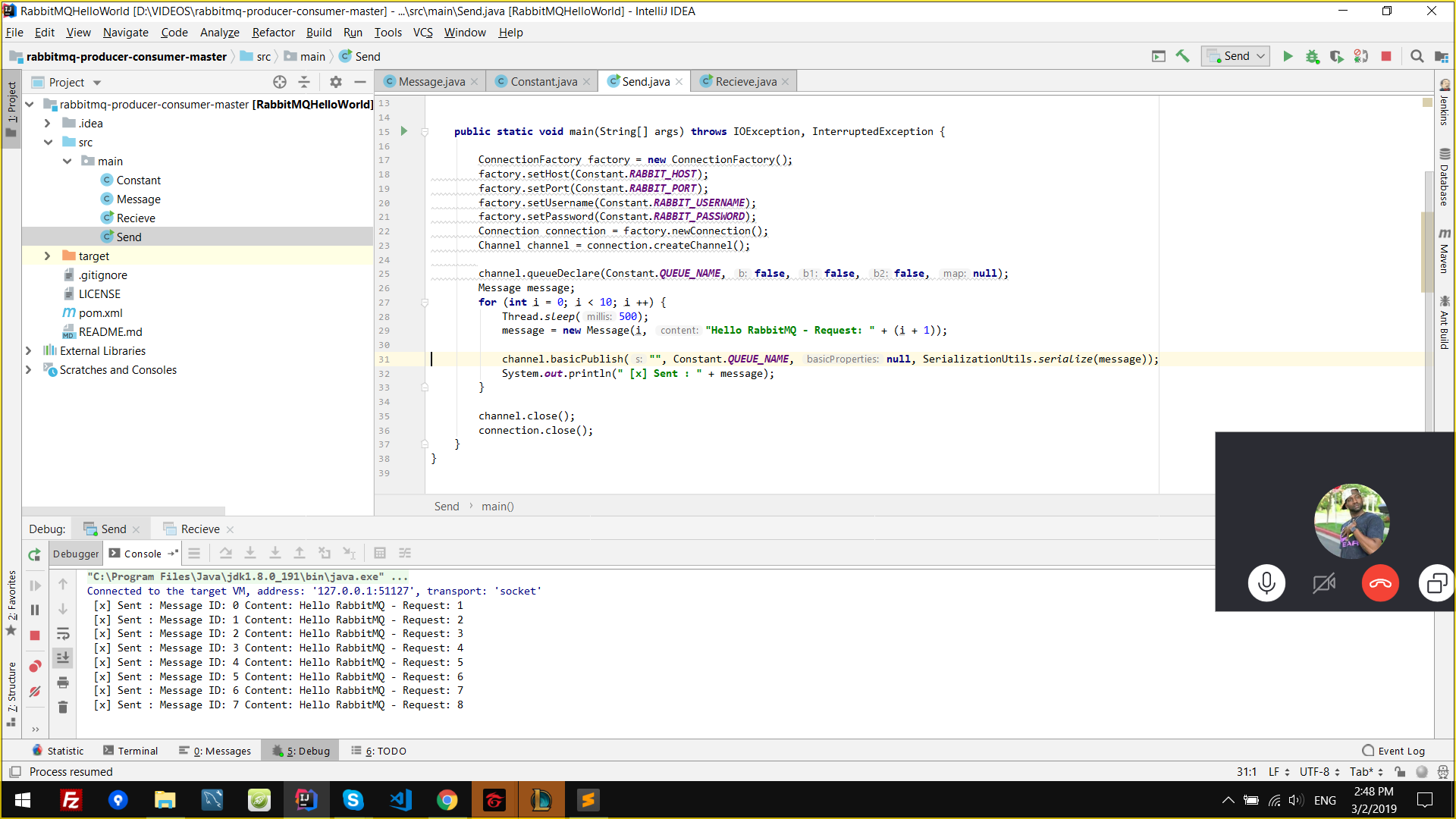


Bên nhận nhận tin nhắn:

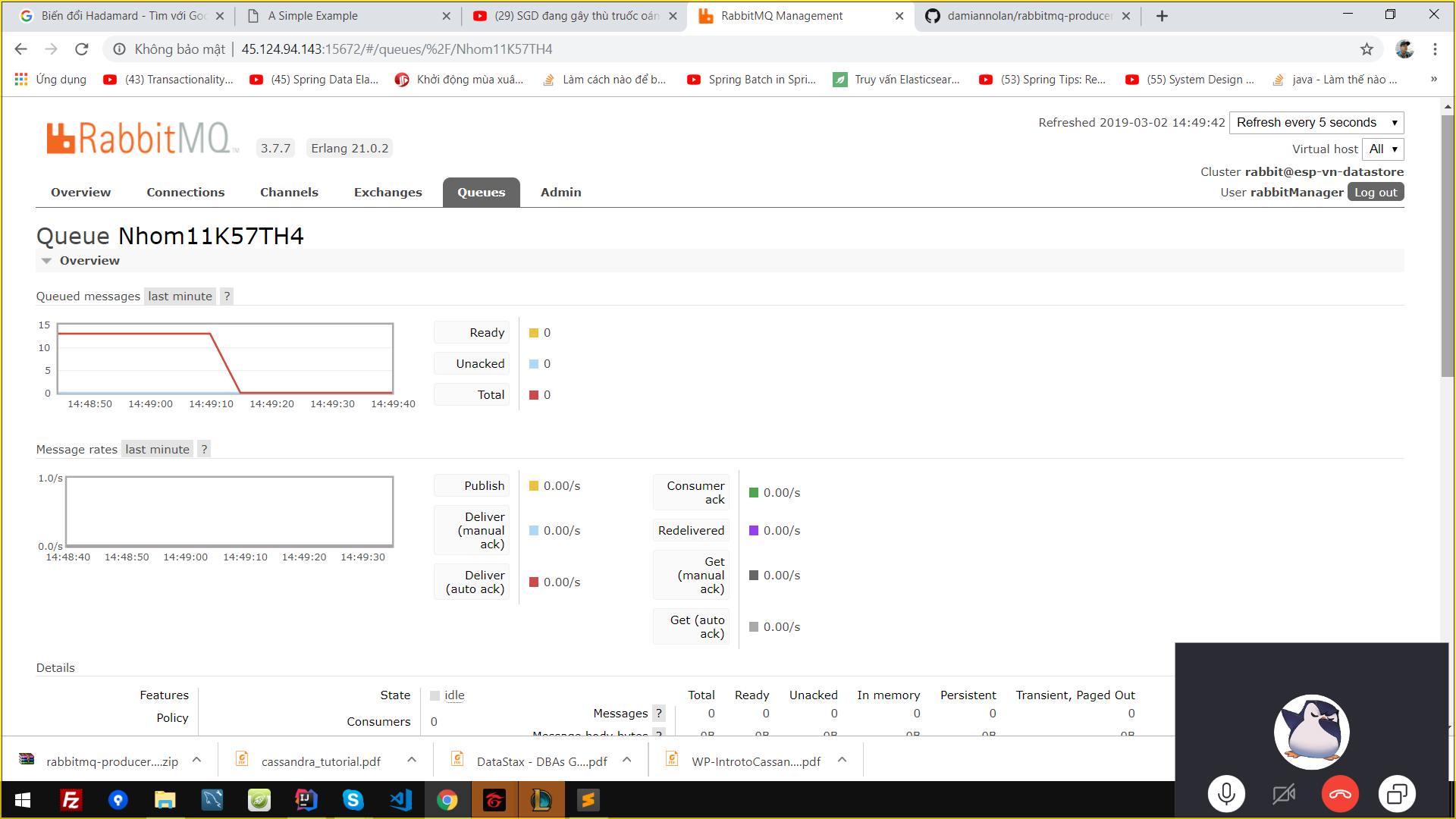




Bên gửi gửi tin nhắn:



Hàng đợi tin nhắn giảm xuống:



Tài Liệu Tham khảo :

https://www.rabbitmq.com/tutorials/tutorial-one-java.html