**Chủ đề: Tìm hiểu và viết một ứng dụng minh hoạ về Spring Kafka**

Người thực hiện: Hoàng Mạnh Dũng – LA 31 PM

Link: <https://github.com/tentaikhoan24/spring-kafka/>

**Hà Nội - 2023**

**Phụ lục**

[A/ Tổng quan: 3](#_Toc143480746)

[I/ MOM - Message Oriented Middleware: 3](#_Toc143480747)

[1/ Khái niệm: 3](#_Toc143480748)

[2/ Danh mục Middleware: 4](#_Toc143480749)

[3/ Ưu điểm 4](#_Toc143480750)

[4/ Nhược điểm: 5](#_Toc143480751)

[II/ Apache Kafka: 6](#_Toc143480752)

[1/ Kafka là gì? 6](#_Toc143480753)

[2/ Kafka hoạt động như thế nào? 6](#_Toc143480754)

[3/ Các khái niệm cơ bản 7](#_Toc143480755)

[4/ Cấu trúc dữ liệu log trong Kafka 8](#_Toc143480756)

[5/ Parsistence data trong Kafa 9](#_Toc143480757)

[6/ Tại sao nên sử dụng Apache Kafka? 9](#_Toc143480758)

[7/ Một vài use case cho Kafka: 10](#_Toc143480759)

[B/ Demo hệ thống đăng ký tài khoản: 10](#_Toc143480760)

[I/ Luồng hoạt động tổng thể 10](#_Toc143480761)

[II/ Cài đặt và demo: 12](#_Toc143480762)

[1. Mô tả hệ thống 12](#_Toc143480763)

[2. Hướng dẫn cài đặt 12](#_Toc143480764)

[2.1. Cài đặt Apache kafka và demo cách hoạt động 12](#_Toc143480765)

[2.2. Tạo kafka producer - accountservice với spring boot 14](#_Toc143480766)

[2.3. Tạo kafka consumer - notificationservice 20](#_Toc143480767)

[2.4. Tạo kafka consumer - statistiscservice 24](#_Toc143480768)

[3. Chạy chương trình và kiểm tra kết quả 26](#_Toc143480769)

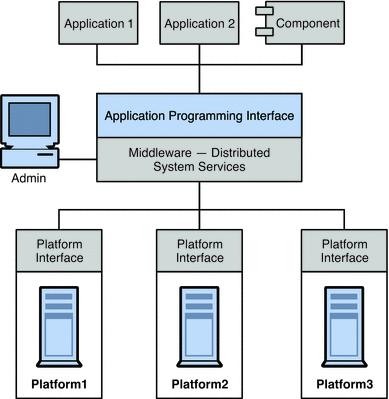
# **A/ Tổng quan:**

## **I/ MOM - Message Oriented Middleware:**

### **1/ Khái niệm:**

Phần mềm trung gian hướng thông báo (MOM) là cơ sở hạ tầng phần mềm hoặc phần cứng hỗ trợ gửi và nhận tin nhắn giữa các hệ thống phân tán. MOM cho phép các mô-đun ứng dụng được phân phối trên các nền tảng không đồng nhất và giảm độ phức tạp của việc phát triển các ứng dụng trải rộng trên nhiều hệ điều hành và giao thức mạng . Phần mềm trung gian tạo ra một lớp truyền thông phân tán, cách ly nhà phát triển ứng dụng khỏi các chi tiết của các hệ điều hành và giao diện mạng khác nhau. API mở rộng trên các nền tảng và mạng khác nhau thường được cung cấp bởi MOM.

Lớp phần mềm trung gian này cho phép các thành phần phần mềm (ứng dụng, Enterprise JavaBeans, servlets và các thành phần khác) được phát triển độc lập và chạy trên các nền tảng mạng khác nhau để tương tác với nhau. Các ứng dụng được phân phối trên các nút mạng khác nhau sử dụng giao diện ứng dụng để giao tiếp. Ngoài ra, bằng cách cung cấp giao diện quản trị, hệ thống ứng dụng ảo mới này có thể trở nên đáng tin cậy và an toàn.



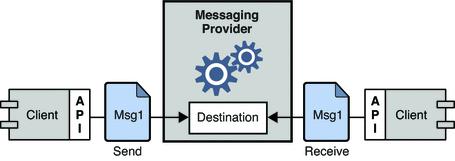
MOM cung cấp các phần tử phần mềm nằm trong tất cả các thành phần giao tiếp của kiến trúc máy khách/máy chủ và thường hỗ trợ các cuộc gọi không đồng bộ giữa ứng dụng khách và máy chủ. MOM làm giảm sự tham gia của các nhà phát triển ứng dụng với sự phức tạp của bản chất chủ-nô của cơ chế máy khách/máy chủ. Xử lý dựa trên thông báo là bắt buộc trong môi trường máy khách/máy chủ sử dụng chương trình đặc biệt gọi là nhà môi giới tin nhắn. Một khách hàng gửi tin nhắn đến nhà môi giới tin nhắn được thiết kế để xử lý nhiều tin nhắn từ nhiều khách

hàng và chuyển tiếp chúng đến ứng dụng máy chủ phù hợp. Phần mềm trung gian tạo ra một lớp truyền thông để cách ly các nhà phát triển khỏi sự phức tạp của các hệ điều hành và giao thức mạng khác nhau.

### **2/ Danh mục Middleware:**

* Cuộc gọi thủ tục từ xa hoặc phần mềm trung gian dựa trên RPC, cho phép các thủ tục trong một ứng dụng gọi các thủ tục trong các ứng dụng từ xa như thể chúng là các cuộc gọi cục bộ. Phần mềm trung gian thực hiện một cơ chế liên kết xác định vị trí các thủ tục từ xa và làm cho chúng có sẵn trong suốt cho người gọi. Theo truyền thống, loại phần mềm trung gian xử lý các chương trình dựa trên thủ tục; bây giờ nó cũng bao gồm các thành phần dựa trên đối tượng.
* Đối tượng yêu cầu đối tượng môi giới hoặc phần mềm trung gian dựa trên ORB, cho phép các đối tượng của ứng dụng được phân phối và chia sẻ trên các mạng không đồng nhất.
* Phần mềm trung gian hướng thư hoặc phần mềm trung gian dựa trên MOM, cho phép các ứng dụng phân tán giao tiếp và trao đổi dữ liệu bằng cách gửi và nhận tin nhắn.

Tất cả các mô hình này làm cho một thành phần phần mềm có thể ảnh hưởng đến hành vi của một thành phần khác qua mạng. Chúng khác nhau ở chỗ phần mềm trung gian dựa trên RPC và ORB tạo ra các hệ thống các thành phần được liên kết chặt chẽ, trong khi các hệ thống dựa trên MOM cho phép ghép các thành phần lỏng lẻo hơn. Trong một hệ thống dựa trên RPC- hoặc ORB, khi một thủ tục gọi một thủ tục khác, nó phải đợi thủ tục được gọi trở lại trước khi nó có thể làm bất cứ điều gì khác. Trong các mô hình nhắn tin đồng bộ này , phần mềm trung gian hoạt động một phần như một siêu liên kết, định vị thủ tục được gọi trên mạng và sử dụng các dịch vụ mạng để truyền tham số hàm hoặc phương thức cho thủ tục và sau đó trả về kết quả.



### **3/ Ưu điểm**

* Lý do chính của việc sử dụng giao thức truyền thông dựa trên tin nhắn bao gồm khả năng lưu trữ (bộ đệm), định tuyến hoặc chuyển đổi tin nhắn trong khi truyền chúng từ người gửi đến người nhận.
  + *Asynchronicity:*
    - Sử dụng hệ thống MOM, khách hàng thực hiện cuộc gọi API để gửi tin nhắn đến đích do nhà cung cấp quản lý. Cuộc gọi gọi các dịch vụ của nhà cung cấp để định tuyến và gửi tin nhắn. Khi nó đã gửi tin nhắn, khách hàng có thể tiếp tục thực hiện các công việc khác, tự tin rằng nhà cung cấp giữ lại tin nhắn cho đến khi một khách hàng nhận được nó. Mô hình dựa trên thông báo, kết hợp với sự trung gian của nhà cung cấp, cho phép tạo ra một hệ thống các thành phần được ghép lỏng lẻo.

* + - MOM bao gồm một danh mục phần mềm giao tiếp liên ứng dụng thường dựa vào việc truyền thông điệp không đồng bộ , trái ngược với kiến trúc phản hồi yêu cầu . Trong các hệ thống không đồng bộ, hàng đợi tin nhắn cung cấp lưu trữ tạm thời khi chương trình đích bận hoặc không được kết nối. Ngoài ra, hầu hết các hệ thống MOM không

đồng bộ cung cấp lưu trữ liên tục để sao lưu hàng đợi tin nhắn. Điều này có nghĩa là người gửi và người nhận không cần kết nối với mạng cùng một lúc ( phân phối không đồng bộ ) và các vấn đề với kết nối không liên tục được giải quyết. Điều đó cũng có nghĩa là nếu ứng dụng người nhận thất bại vì bất kỳ lý do gì, người gửi có thể tiếp tục không bị ảnh hưởng, vì các tin nhắn họ gửi sẽ chỉ tích lũy trong hàng đợi tin nhắn để xử lý sau khi người nhận khởi động lại.

* + *Định tuyến:* Nhiều triển khai phần mềm trung gian hướng thông báo phụ thuộc vào hệ thống hàng đợi tin nhắn . Một số triển khai cho phép logic định tuyến được cung cấp bởi chính lớp nhắn tin, trong khi các cài đặt khác phụ thuộc vào ứng dụng khách để cung cấp thông tin định tuyến hoặc cho phép kết hợp cả hai mô hình. Một số triển khai sử dụng các mô hình phân phối quảng bá hoặc phát đa hướng.
  + *Chuyển đổi:* Trong một hệ thống phần mềm trung gian dựa trên tin nhắn, tin nhắn nhận

được tại đích không cần phải giống hệt với tin nhắn được gửi ban đầu. Một hệ thống MOM với trí thông minh tích hợp có thể chuyển đổi các thông điệp trên đường để phù hợp với yêu cầu của người gửi hoặc của người nhận. Cùng với các phương tiện định tuyến và phát/phát đa hướng , một ứng dụng có thể gửi tin nhắn ở định dạng gốc của riêng mình và hai hoặc nhiều ứng dụng khác có thể nhận được một bản sao của tin nhắn ở định dạng riêng của chúng. Nhiều hệ thống MOM hiện đại cung cấp các công cụ chuyển đổi thông điệp (hoặc ánh xạ) tinh vi cho phép các lập trình viên chỉ định các quy tắc chuyển đổi áp dụng cho thao tác kéo và thả GUI đơn giản.

* Một ưu điểm khác của việc nhắn tin qua trung gian của nhà cung cấp dịch vụ nhắn tin giữa các khách hàng là bằng cách thêm giao diện quản trị, bạn có thể theo dõi và điều chỉnh hiệu suất. Do đó, các ứng dụng khách được giải quyết hiệu quả mọi vấn đề ngoại trừ việc gửi, nhận và xử lý tin nhắn. Tùy thuộc vào mã thực hiện hệ thống MOM và tùy thuộc vào quản trị viên

để giải quyết các vấn đề như khả năng tương tác, độ tin cậy, bảo mật, khả năng mở rộng và hiệu suất.

### **4/ Nhược điểm:**

* Nhược điểm chính của nhiều hệ thống phần mềm trung gian hướng thông báo là chúng yêu cầu một thành phần bổ sung trong kiến trúc , tác nhân chuyển tin nhắn (nhà môi giới tin nhắn). Như với bất kỳ hệ thống nào , việc thêm một thành phần khác có thể dẫn đến giảm hiệu suất và độ tin cậy, và cũng có thể làm cho toàn bộ hệ thống trở nên khó khăn và tốn kém hơn để duy trì.
* Ngoài ra, nhiều liên lạc giữa các ứng dụng có một khía cạnh đồng bộ nội tại , với người gửi đặc biệt muốn chờ trả lời tin nhắn trước khi tiếp tục (xem điện toán thời gian thực và gần thời gian thực cho các trường hợp cực đoan). Bởi vì giao tiếp dựa trên thông điệp vốn có chức năng không đồng bộ, nên nó có thể không phù hợp trong các tình huống như vậy. Điều đó nói

rằng, hầu hết các hệ thống MOM đều có các phương tiện để nhóm một yêu cầu và phản hồi dưới dạng một giao dịch giả đồng bộ.

* Với một hệ thống nhắn tin đồng bộ, chức năng gọi sẽ không trở lại cho đến khi chức năng được gọi đã hoàn thành nhiệm vụ. Trong một hệ thống không đồng bộ được ghép lỏng lẻo , máy khách gọi có thể tiếp tục tải công việc cho người nhận cho đến khi tài nguyên cần thiết để xử lý công việc này bị cạn kiệt và thành phần được gọi là thất bại. Tất nhiên, những điều kiện này có thể được giảm thiểu hoặc tránh bằng cách theo dõi hiệu suất và điều chỉnh luồng thông báo, nhưng đây là công việc không cần thiết với hệ thống nhắn tin đồng bộ. Điều quan trọng là phải hiểu những lợi thế và trách nhiệm của từng loại hệ thống. Mỗi hệ thống phù hợp cho các loại nhiệm vụ khác nhau. Đôi khi, cần phải có sự kết hợp của hai loại hệ thống để có được hành vi mong muốn.

## **II/ Apache Kafka:**

### **1/ Kafka là gì?**

Kafka là nền tảng streaming phân tán, có thể mở rộng và là sản phẩm mã nguồn mở.

Dự án Kafka ban đầu được phát triển bởi Linkedin sau đó trở thành dự án Apache mã nguồn

mở vào năm 2011. Kafka được viết bằng ngôn ngữ Scala và Java. Nó được viết ra nhằm mục

đích cung cấp một nền tảng mà có độ trễ thấp và thông lượng cao cho việc xử lý các nguồn

cấp dữ liệu theo thời gian thực.

### **2/ Kafka hoạt động như thế nào?**

Kafka được xây dựng dựa trên mô hình publish/subcribe phân tán (distributed

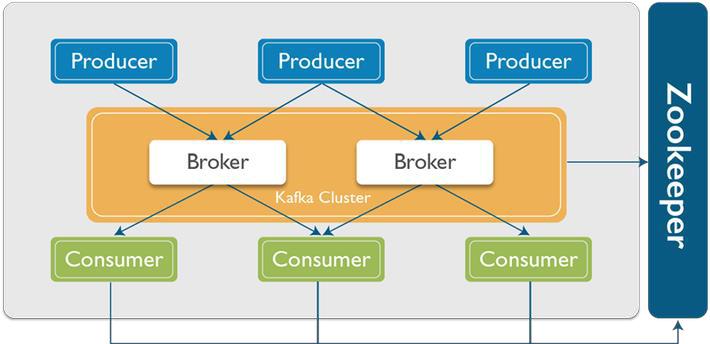
messaging system). Bên pulbic dữ liệu được gọi là producer, bên subscribe nhận dữ liệu theo

topic được gọi là consumer. Kafka có khả năng truyền một lượng lớn message theo thời gian

thực, trong trường hợp bên nhận chưa nhận message vẫn được lưu trữ sao lưu trên một hàng

đợi và cả trên ổ đĩa bảo đảm an toàn. Đồng thời nó cũng được replicate trong cluster giúp

phòng tránh mất dữ liệu.



*Hình 1: Một cấu trúc kafka đơn giản*

### **3/ Các khái niệm cơ bản**

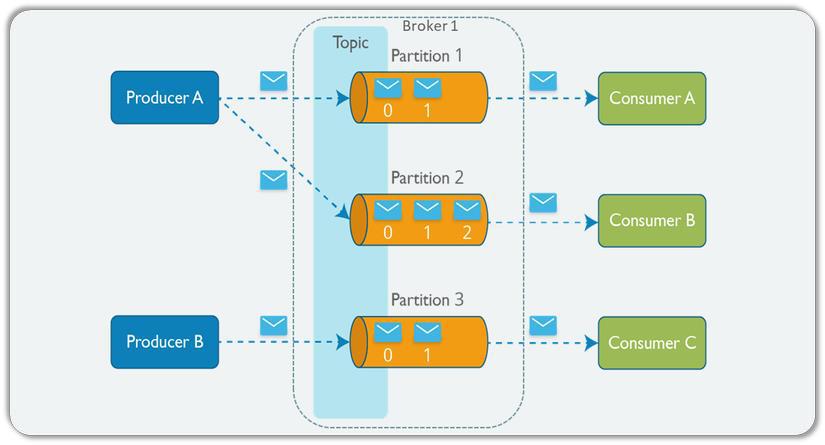
Kafka có thể hiểu là một hệ thống logging để lưu lại các trạng thái của hệ thống đề phòng tránh mất thông tin.

**Định nghĩa trên được giải thích bằng các khái niệm sau:**

* *PRODUCER*: Kafka lưu, phân loại message theo topic, sử dụng producer để publish message vào các topic. Dữ liệu được gửi đển partition của topic lưu trữ trên Broker.
* *CONSUMER***:** Kafka sử dụng consumer để subscribe vào topic, các consumer được định danh bằng các group name. Nhiều consumer có thể cùng đọc một topic.
* *TOPIC*: Dữ liệu truyền trong Kafka theo topic, khi cần truyền dữ liệu cho các ứng dụng khác nhau thì sẽ tạo ra cá topic khác nhau.
* *PARTITION*: Đây là nơi dữ liệu cho một topic được lưu trữ. Một topic có thể có một hay nhiều partition. Trên mỗi partition thì dữ liệu lưu trữ cố định và được gán cho một ID gọi là offset. Trong một Kafka cluster thì một partition có thể replicate (sao chép)

ra nhiều bản. Trong đó có một bản leader chịu trách nhiệm đọc ghi dữ liệu và các bản còn lại gọi là follower. Khi bản leader bị lỗi thì sẽ có một bản follower lên làm leader thay thế. Nếu muốn dùng nhiều consumer đọc song song dữ liệu của một topic thì topic đó cần phải có nhiều partition.

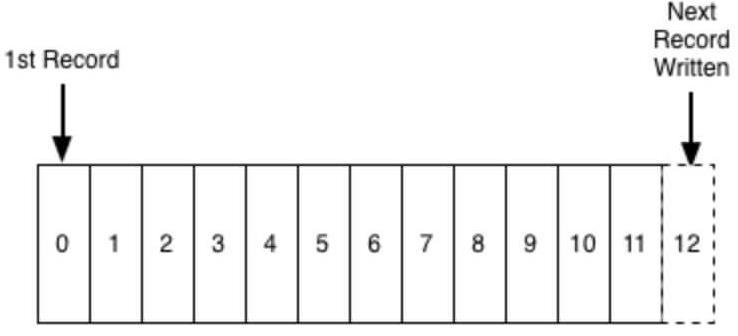
* *BROKER*: Kafka cluster là một set các server, mỗi một set này được gọi là 1 broker
* *ZOOKEEPER*: được dùng để quản lý và bố trí các broker.



*Hình 2: Cấu trúc kafka chi tiết*

### **4/ Cấu trúc dữ liệu log trong Kafka**

Chìa khóa chính dẫn tới khả năng mở rộng và hiệu suất của kafka chính là log. Thường thì các developer khi mới tiếp cận kafka cảm thấy khá rối khi lần đầu tiên nghe đến “**log**”, bởi vì chúng ta thường hiểu “log” chính là thuật nghữ được sử dụng trong log ứng dụng. Tuy nhiên, những gì mình đang nói ở đây, là cấu trúc dữ liệu log. Log là một cấu trúc dữ liệu có thứ tự nhất quán mà chỉ hỗ trợ dạng nối thêm (append). Bạn không thể chỉnh sửa hay xóa các records từ nó. Nó được đọc từ trái sang phải và được đảm bảo thứ tự các item.



*Hình 3: Cấu trúc dữ liệu log*

Một nguồn dữ liệu sẽ ghi message vào log và một hoặc nhiều consumer khác sẽ đọc message từ log tại thời điểm họ lựa chọn.

Mỗi entry trong log được định danh bởi một con số gọi là offset, hay nói một cách dễ hiểu hơn, offset giống như chỉ số tuần tự trong một array vậy.

Vì chuỗi/offset chỉ có thể được duy trì trên từng node/broker cụ thể và không thể được duy trì đối với toàn bộ cluster, do đó Kafka chỉ đảm bảo sắp xếp thứ tự dữ liệu cho mỗi partition.

### **5/ Parsistence data trong Kafa**

Kafka lưu trữ tất cả message vào disk (không hề lưu trên RAM) và được sắp xếp có thứ tự trong cấu trúc log cho phép kafka tận dụng tối đa khả năng đọc và ghi lên disk một cách tuần tự.

Nó là một cách lựa chọn khá phổ biến để lưu trữ dữ liệu trên disk mà vẫn có thể sử dụng tối đa hóa hiệu năng, có một số lý do chính dưới đây:

* Kafka có một giao thức mà nhóm các message lại với nhau. Điều này cho phép các request network nhóm các message lại với nhau, giúp giảm thiểu chi phí sử dụng tài nguyên mạng, server, gom các message lại thành một cục và consumer sẽ tìm nạp một khối message cùng một lúc – do đó sẽ giảm tải disk cho hệ điều hành.
* Kafka phụ thuộc khá nhiều vào pagecache của hệ điều hành cho việc lưu trữ dữ liệu, sử dụng RAM trên máy một cách hiệu quả.
* Kafka lưu trữ các messages dưới định dạng nhị phân xuyên suốt quá trình (producer > broker > consumer), làm cho nó có thể tận dụng tối ưu hóa khả năng zero-copy. Nghĩa là khi hệ điều hành copy dữ liệu từ pagecache trực tiếp sang socket, hoàn toàn bỏ qua ứng dụng trung gian là kafka.
* Đọc/ghi dữ liệu tuyến tính trên disk nhanh. Vấn đề làm cho disk chậm hiện nay thường là do quá trình tìm kiếm trên disk nhiều lần. Kafka đọc và ghi trên disk tuyến tính, do đó nó có thể tận dụng tối đa hóa hiệu suất trên disk.

### **6/ Tại sao nên sử dụng Apache Kafka?**

Kafka là dự án opensoure, đã được đóng gói hoàn chỉnh, khả năng chịu lỗi cao, hiệu năng rất tốt và dễ dàng mở rộng mà không cần dừng hệ thống.

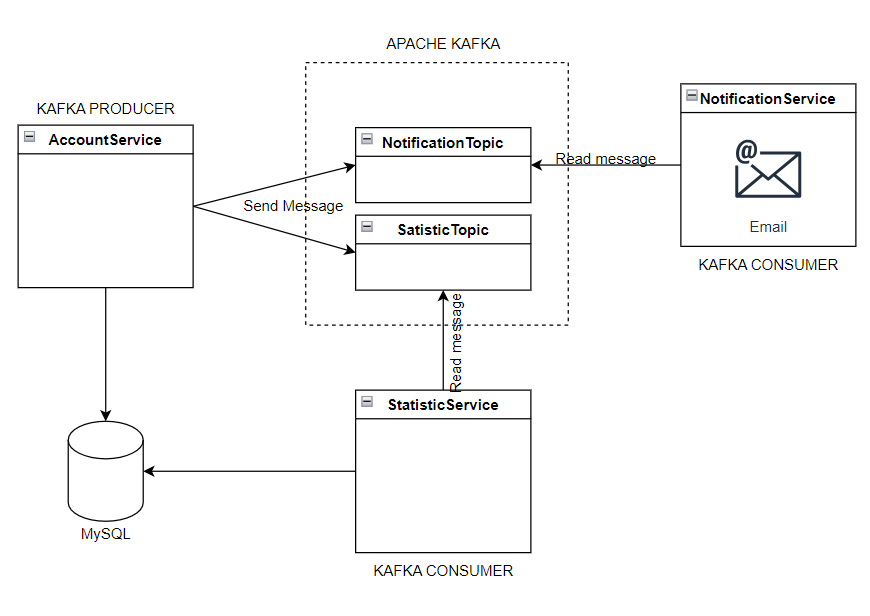
Kafka thật sự đáng tin cậy, có khả năng lưu trữ lượng dữ liệu lớn nên nó đang dần được thay thế cho hệ thống message truyền thống.

### **7/ Một vài use case cho Kafka:**

* Sử dụng như một hệ thống message queue thay thế cho ActiveMQ hay RabbitMQ
* *Website Activity Monitoring:* theo dõi hoạt động của website
* *Stream Processing:* Kafka là một hệ thống rất thích hợp cho việc xử lý dòng dữ liệu trong thời gian thực. Khi dữ liệu của một topic được thêm mới ngay lập tức được ghi vào hệ thống và truyền đến cho bên nhận. Ngoài ra Kafka còn là một hệ thống có đặc tính duribility dữ liệu có thể được lưu trữ an toàn cho đến khi bên nhận sẵn sàng nhận nó.
* *Log Aggregation:* tổng hợp log
* *Metrics Collection:* thu thập dữ liệu, tracking hành động người dùng như các thông số như page view, search action của user sẽ được publish vào một topic và sẽ được xử lý sau
* *Event-Sourcing:* Lưu lại trạng thái của hệ thống để có thể tái hiện trong trường hợp system bị down.

# **B/ Demo hệ thống đăng ký tài khoản:**

## **I/ Luồng hoạt động tổng thể**



## **II/ Cài đặt và demo:**

### **1. Mô tả hệ thống**

- Xây dựng hệ thống đăng ký tài khoản, khi đăng ký thành công sẽ được gửi thông báo qua email

- Hệ thống bao gồm:

+ Apache kafka: nhận message

+ 1 producer – account service chịu trách nhiệm đăng ký tài khoản, tạo topic trên kafka và gửi message lên topic khi đăng ký thành công

+ 2 consumer: notification service – đọc message trong topic notificaiotn để nhận thông tin của người đăng ký tài khoản và gửi email đến email đăng ký tài khoản. statistic service – đọc message trong topic statistic để nhận thông tin như email, message và thời gian đăng ký vào database

### **2. Hướng dẫn cài đặt**

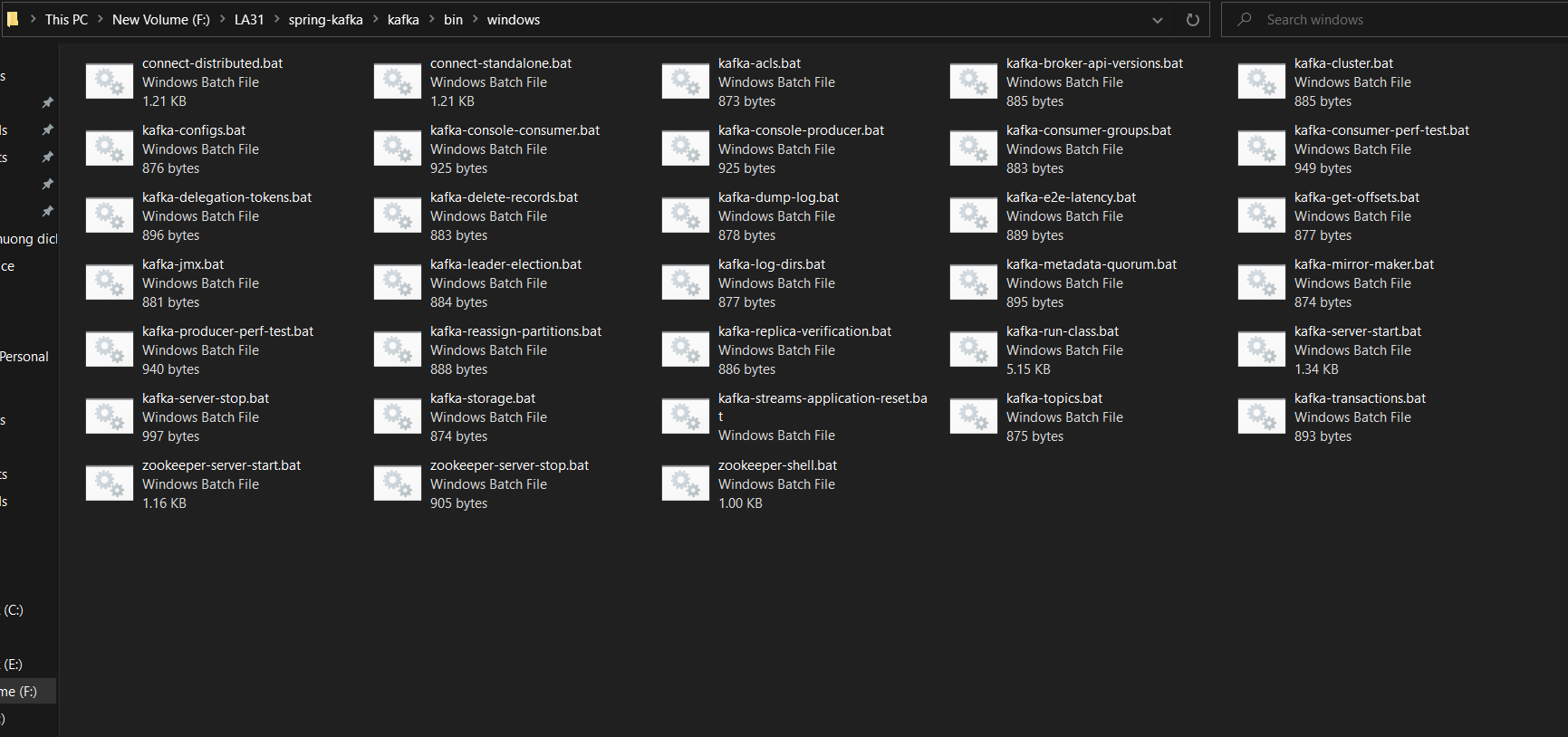
### **2.1. Cài đặt Apache kafka và demo cách hoạt động**

- Yêu cầu cài đặt JDK 8 +

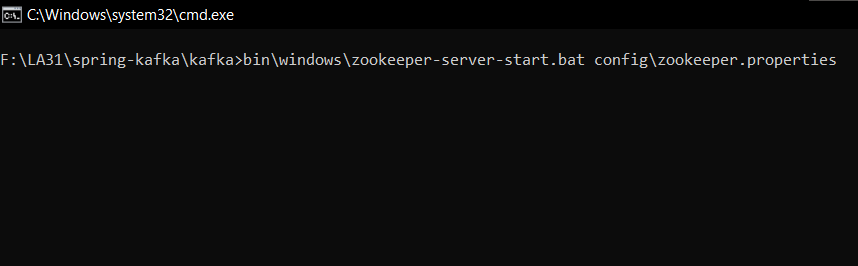
- Download apache kafka tại <https://kafka.apache.org/downloads>

- Giải nén thư mục và kiểm tra các lệnh chạy trên windows tại thư mục bin/window

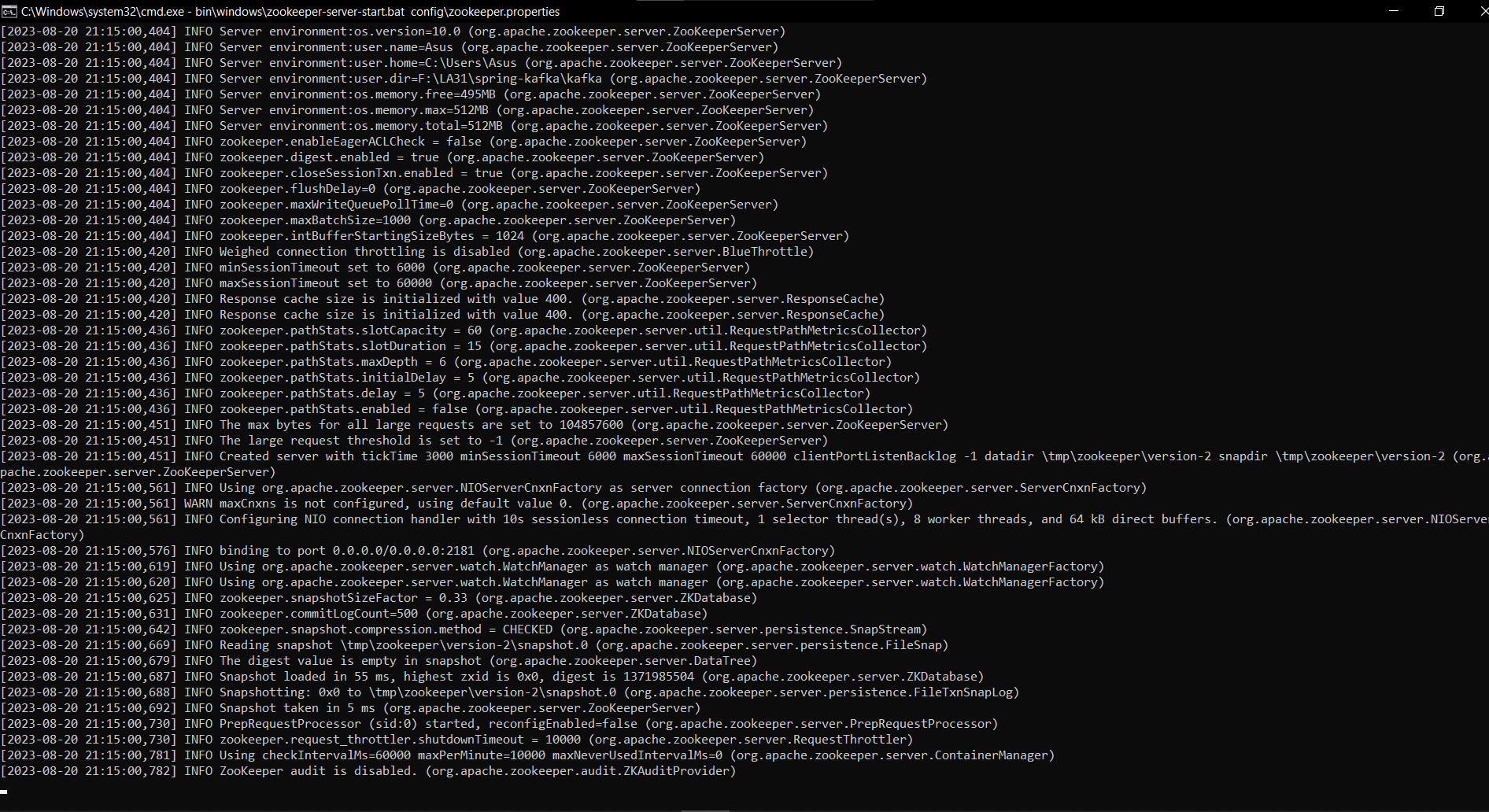
- Nếu chạy lên linux hoặc macos thì các lệnh ngay trong thư mục bin



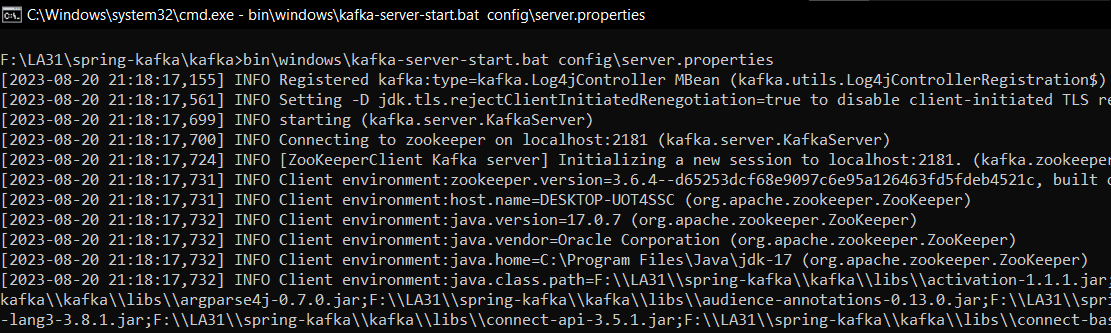
- Khởi chạy zookeeper – server quản lý apache kafka tại folder kafka sau khi đã giải nén bằng lệnh [bin\windows\zookeeper-server-start.bat config\zookeeper.properties] với file config có sẵn trong thư mục config/ zookeeper.properties



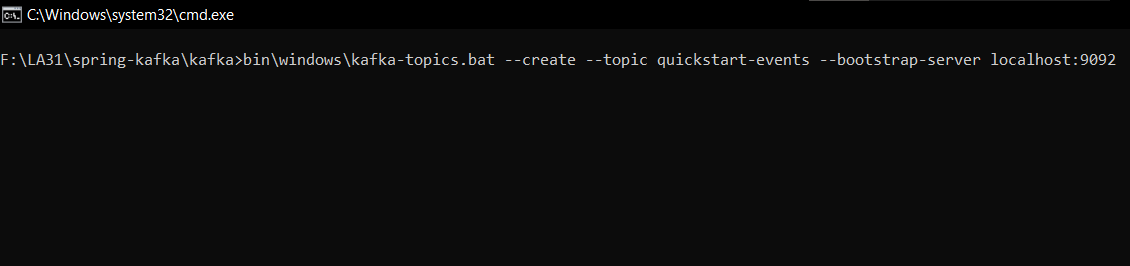
- Sau khi khởi chạy thành công

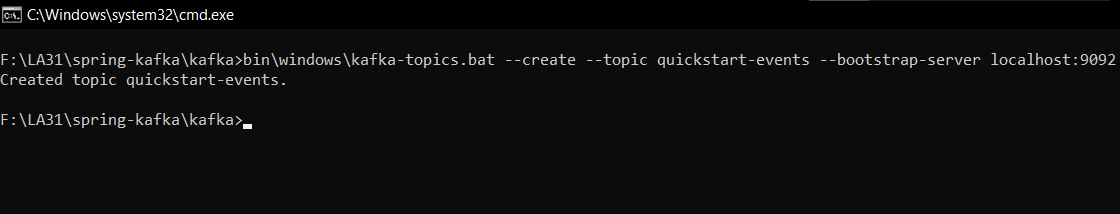


- Khởi chạy server kafka – broker server để kết nối tới zookeeper tại terminal mới với lệnh [bin\windows\kafka-server-start.bat config\server.properties] với file config có sẵn tại config\server.properties

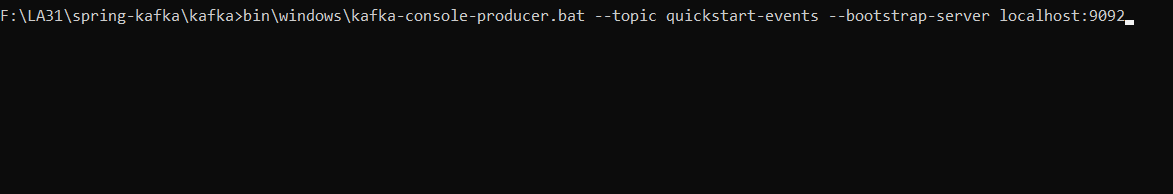


- Tạo topic quickstart-events để nhận message event và kết nối tới zookeeper với cổng mặc định là 9092 bằng lệnh [bin\windows\kafka-topics.bat --create --topic quickstart-events --bootstrap-server localhost:9092]

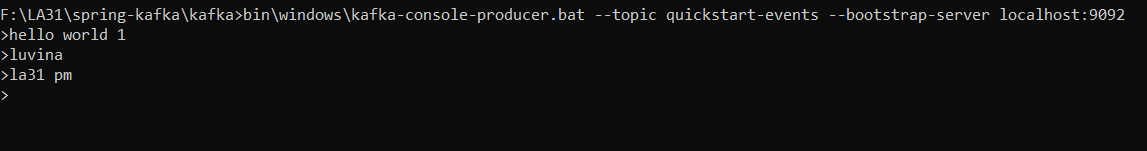




- Tạo 1 producer để gửi event tới topic quickstart-event đã tạo ở trên với lệnh [bin\windows\kafka-console-producer.bat --topic quickstart-events --bootstrap-server localhost:9092]

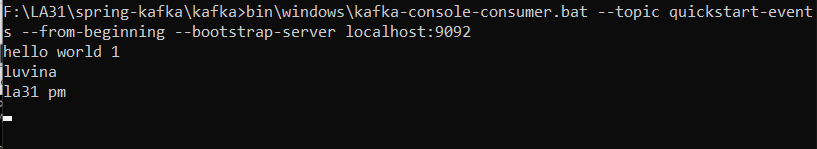


- Tạo các event string đơn giản từ producer tới topic quickstart-events

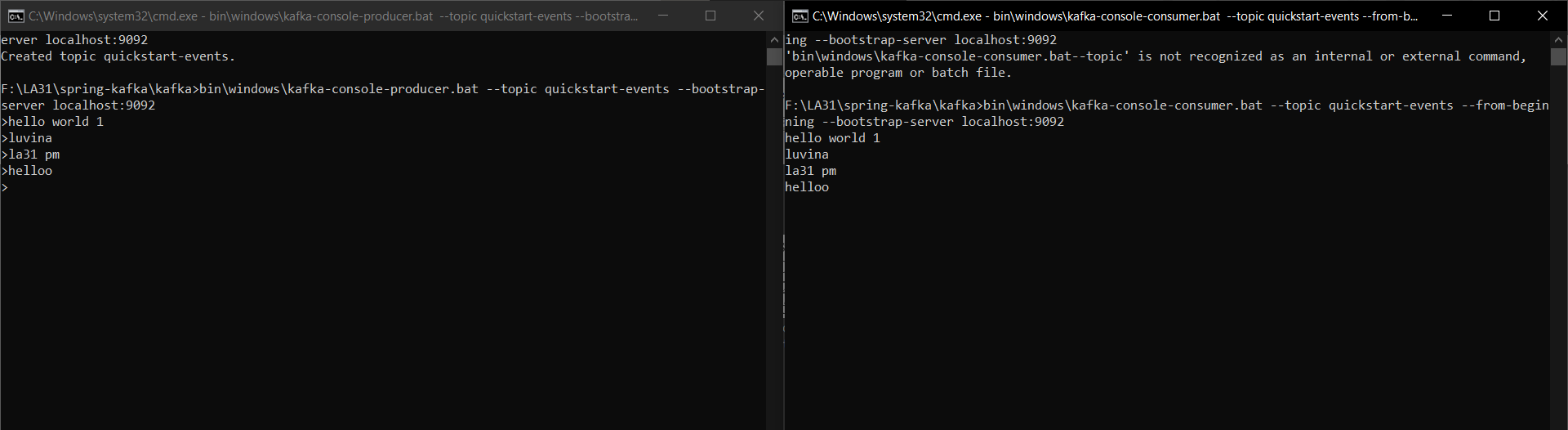


- Tạo consumer để đọc event trong topic quickstart-events với lệnh [--topic quickstart-events --from-beginning --bootstrap-server localhost:9092] với các tham số như --from-beginning để đọc tất cả event từ vị trí đầu tiên, localhost:9092 là địa chỉ của broker chứa topic đã tạo

- Consumer đã đọc được các event đã gửi đi trước đó từ producer



- Consumer đọc các event real time nhưng vẫn có 1 độ chễ nhất định khi phải tạo event từ producer và gửi lên topic và sau đó mới được consumer đọc



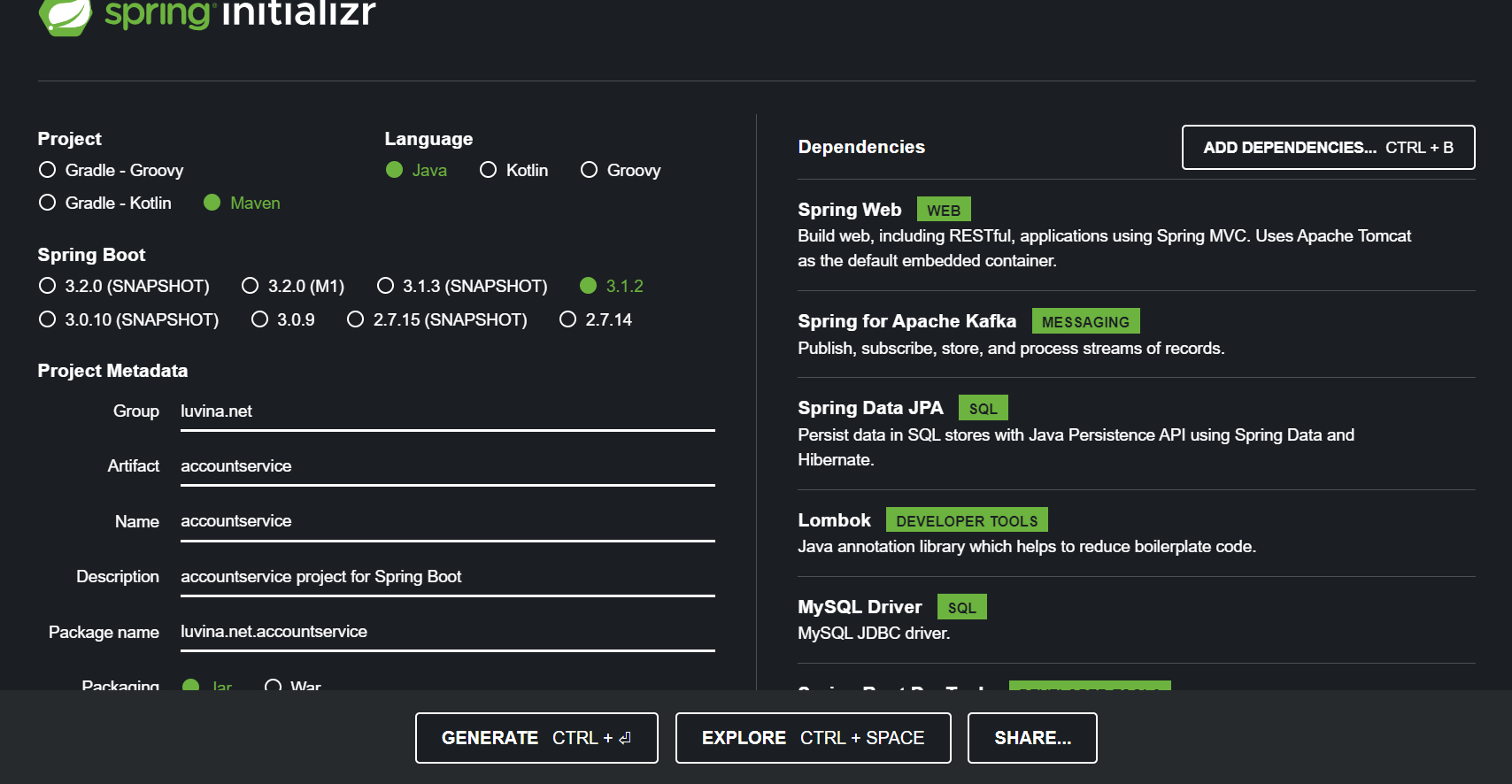
### 2.2. Tạo kafka producer - accountservice với spring boot

2.2.1. Khởi tạo project

- Tạo accountservice với vai trò là 1 producer

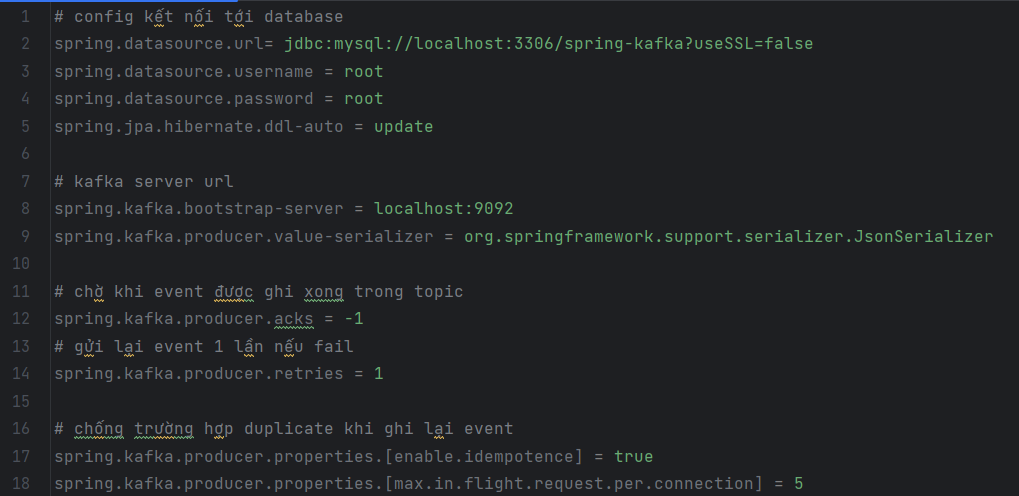
- Tạo project spring boot tại <https://start.spring.io/>

- Chọn các Dependencies cần thiết như spring web, apache kafka,…



2.2.2. Cấu hình

- Mở project và cấu hình tại application.properties



2.2.3. Tạo topic

- Tạo bean – topic trong file application qua phương thức new topic trong thư viện của apache kafka

- Ta sẽ tạo 2 topic là notification và statistic



2.3.4. Tạo các model

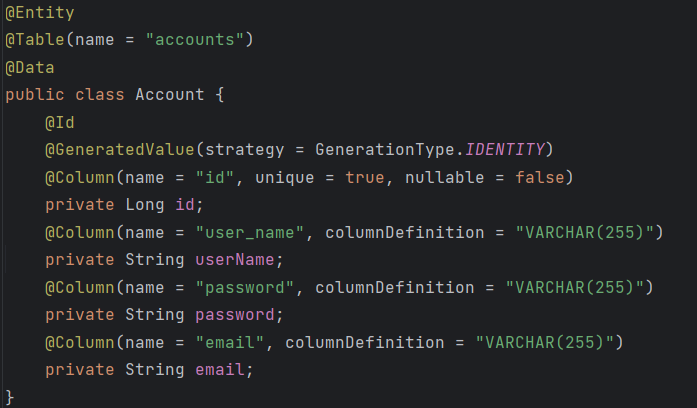
- Sử dụng annotation @Data để tự động tạo các phương thức như: contructer, getter, setter

- Tạo account model chứa thông tin đăng kí từ client

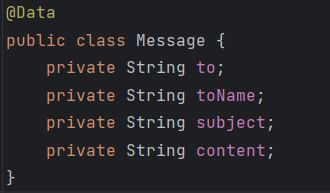
+ Sử dụng annotation @Entity để khai báo đây là 1 entity tương ứng với 1 bảng trong database

+ @Table: khai báo tên table trong database

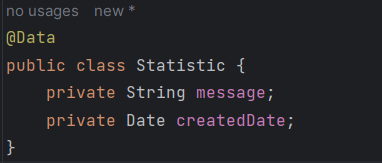
+ @Column: Khai báo tên cột trong bảng và các thuộc tính khác



- Tạo message model chứa các thông tin như: message, subject, content..



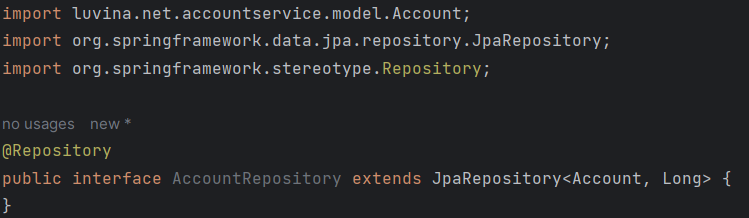
- Tạo statistic model chứa thông tin của account đã tạo



2.3.5. Tạo repository

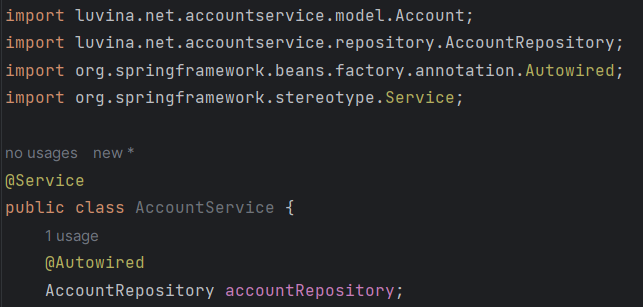
- Tạo AccountRepository để thao tác với database thông qua JpaRepository

@Repository: khai báo đây là 1 repository

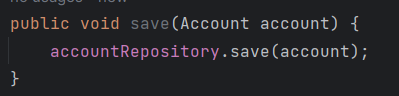


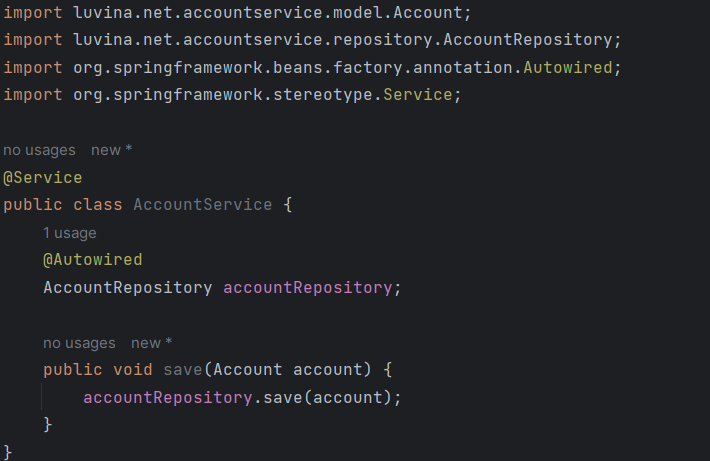
2.3.5. Tạo service chứa các phương thức thao tác với database

@Service: khai báo đây là 1 service



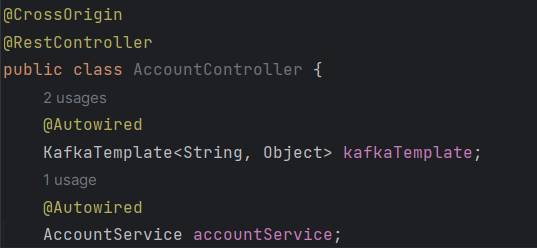
- Tạo phương thức để lưu account vào database thông qua phương thức save có sẵn của jpa



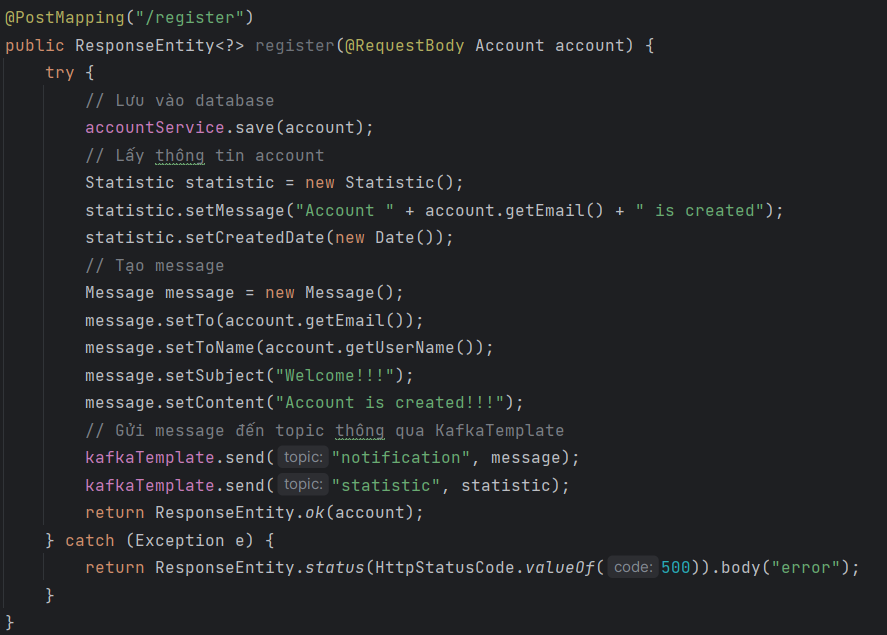


2.3.6. Tạo controller

@ Autowired KafkaTemplate và AccountService để sử dụng các phương thức của chúng



- Tạo api đăng ký account và gửi message đến các topic notification và statistic



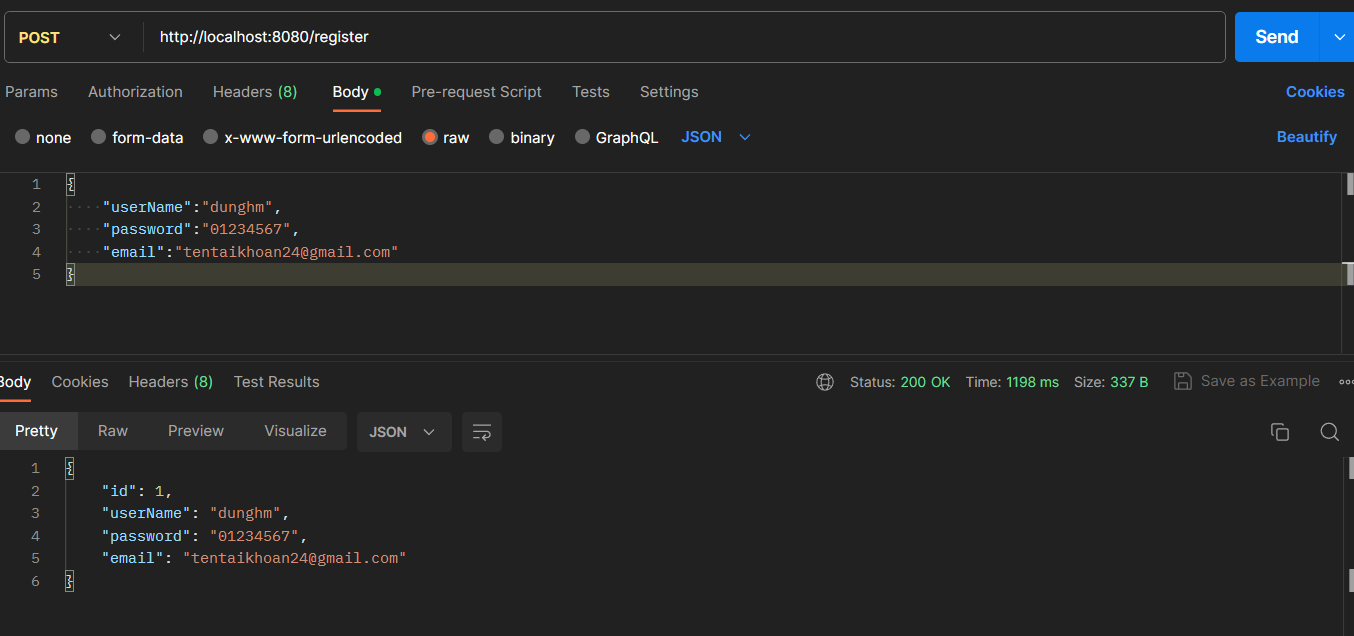
2.3.7. Test controller và đọc message event trong topic

- Mở topic trên terminal với các lệnh tương ứng

[bin\windows\kafka-console-consumer.bat --topic notification --from-beginning --bootstrap-server localhost:9092] – mở topic notification

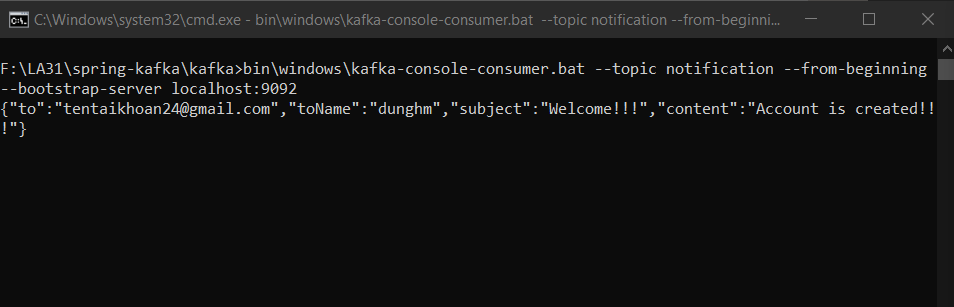
[bin\windows\kafka-console-consumer.bat --topic statistic --from-beginning --bootstrap-server localhost:9092] – mở topic statistic

- Test trên postman với url <http://localhost:8080/register> và method là POST

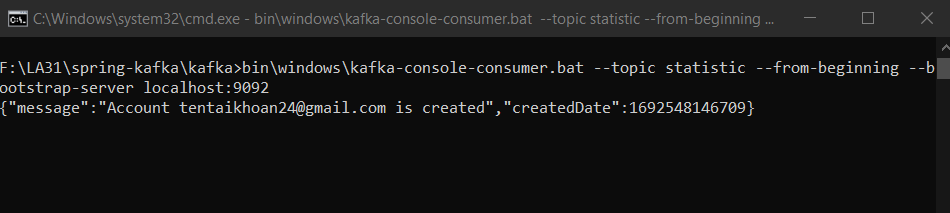


- Kiểm tra 2 topic đã nhận được message event hay chưa

+ Topic notification



+ Topic statistic



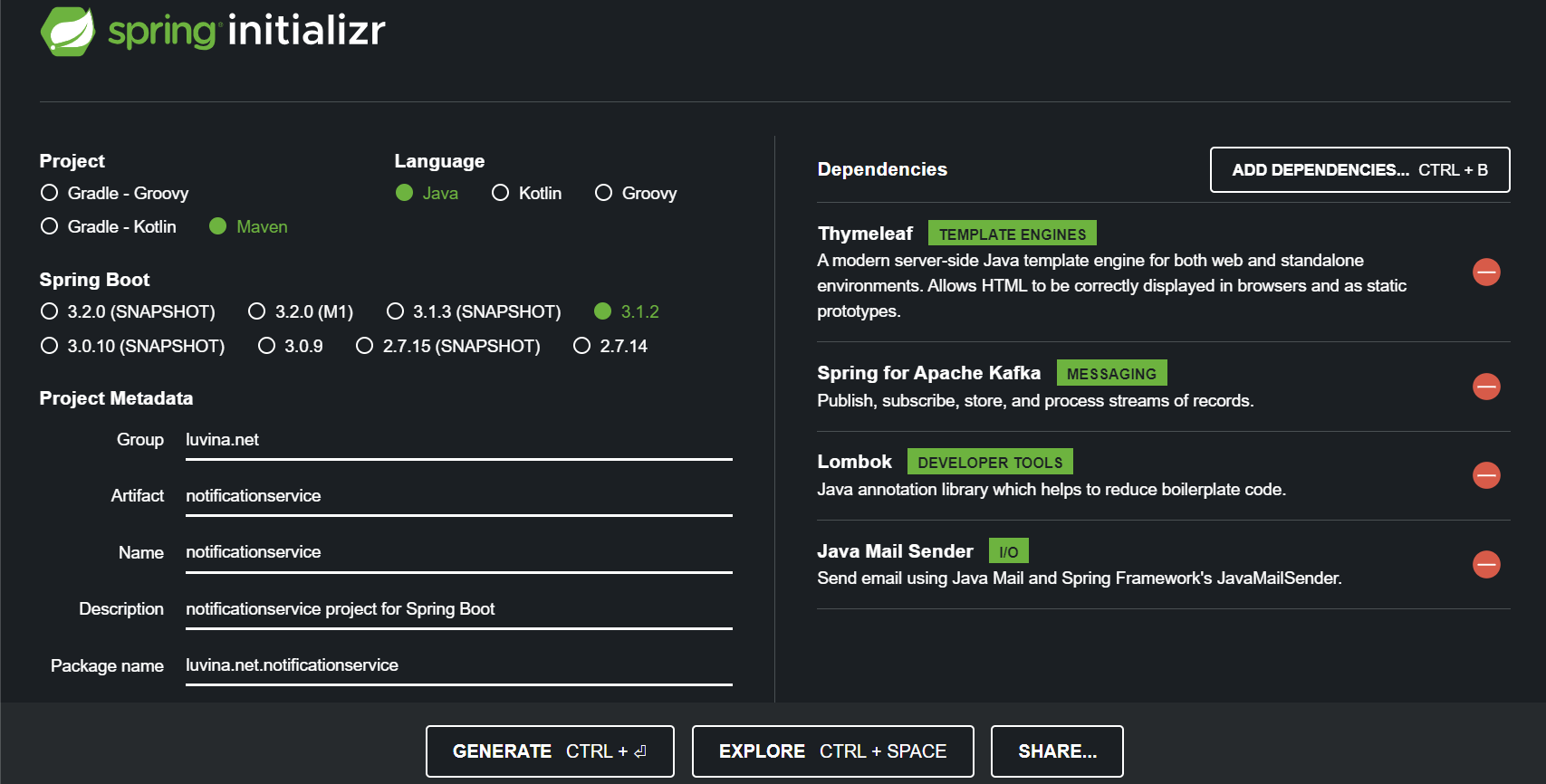
- 2 topic đều đã nhận được message gửi đi từ producer là accountservice

### 2.3. Tạo kafka consumer - notificationservice

- Tạo notificationservice với vai trò là 1 consumer

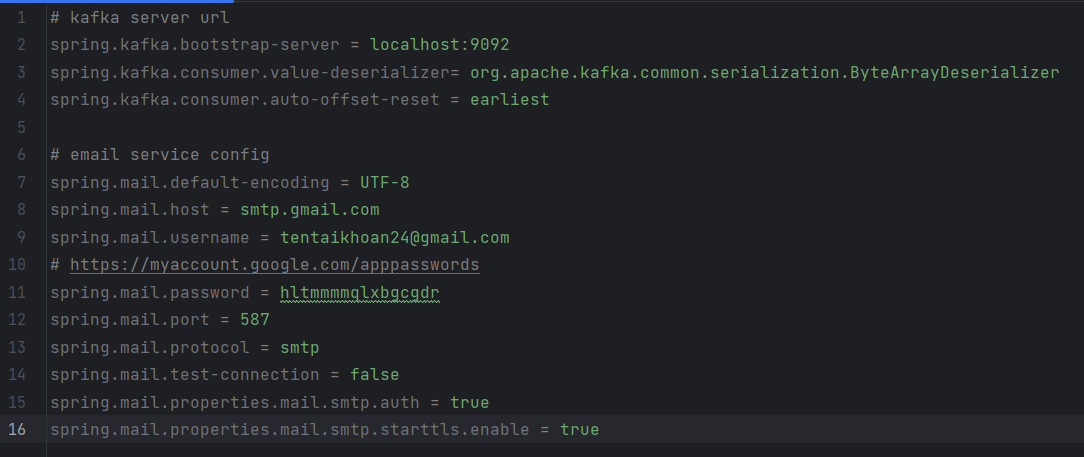
- Tạo project spring boot tại <https://start.spring.io/>

- Chọn các Dependencies cần thiết apache kafka, lombok



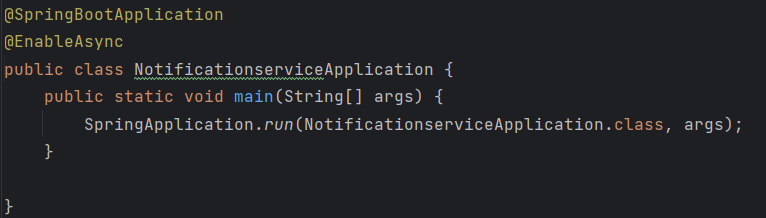
2.3.2. Cấu hình

- Cấu hình cho notificationserice tại file application.properties



2.3.3. Cấu hình file NotificationserviceApplication

- @EnableAsync để tạo luồng mới khi gửi email



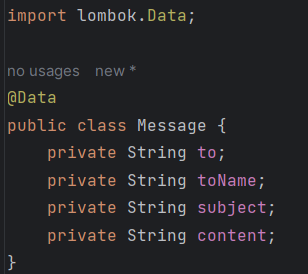
- Thêm @Bean với kiểu dữ liệu là JsonbMessageConverter để nhận và convert dữ liệu từ message event trong topic



2.3.4. Tạo model message

- Tạo model message với các thành phần tương ứng với model message đã tạo trong producer–accountservice

- Sử dụng @Data để tự động tạo contructer và các phương thức setter, getter



2.3.5. Tạo EmailService

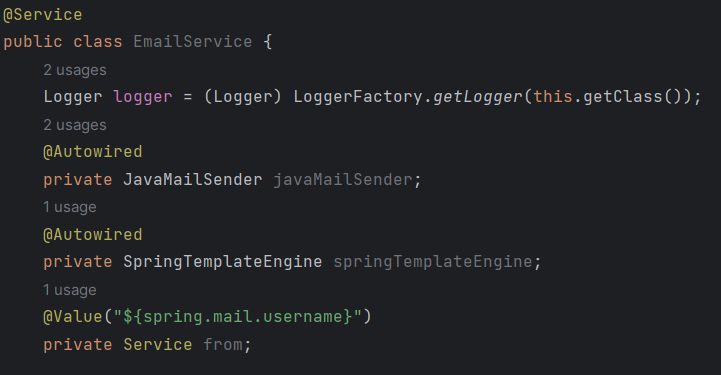
- Tạo emailservice để gửi email qua gmail đã được config

- Tạo logger đẻ ghi log lên terminal

- Sử dụng JavaMailSender để gửi mail

- SpringTemplateEngine để sử dụng template cho email

- @Value("${spring.mail.username}") để lấy dữ liệu từ file config application.properties



- @Async: tạo luồng mới để gửi email, để xử lý bất đồng bộ tránh trường hợp đợi sự kiện gửi mail trên gmail server

- MimeMessage là 1 đối tợng JavaMail, dùng để thiết lập các thuộc tính người gửi, người nhận, tiêu đề, nội dung

- MimeMessageHelper dùng để thiết lập bộ kí tự UTF-8 có hỗ trợ tiếng Việt

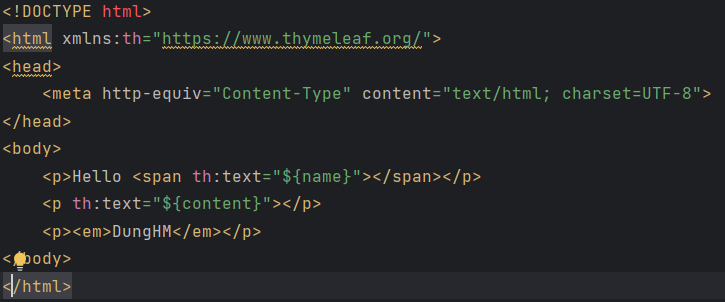
- Context là 1 đối tượng dùng để truyền dữ liệu

Vào html là template đã được tạo trong resource\templates\welcome.html

- MimeMessageHelper hỗ trợ các phương thức gủi mail



- Template welcome.html



2.3.6. Tạo message service

- Tạo service tự động đọc tất cả các message event từ topic được chỉ định

- @Service: để đánh dấu đây là 1 service

- @Autowired để sử dụng EmailService đã tạo trước đó

- @KafkaListener: đọc message event từ topic – id: tạo id cho consumer nếu chạy nhiều consumer 1 lúc sẽ có chung 1 id, topics: topic được chỉ định để đọc message

- Message được đọc sẽ tự động convert sang đối tượng Message và gọi đến hàm sendEmail của EmailMessage

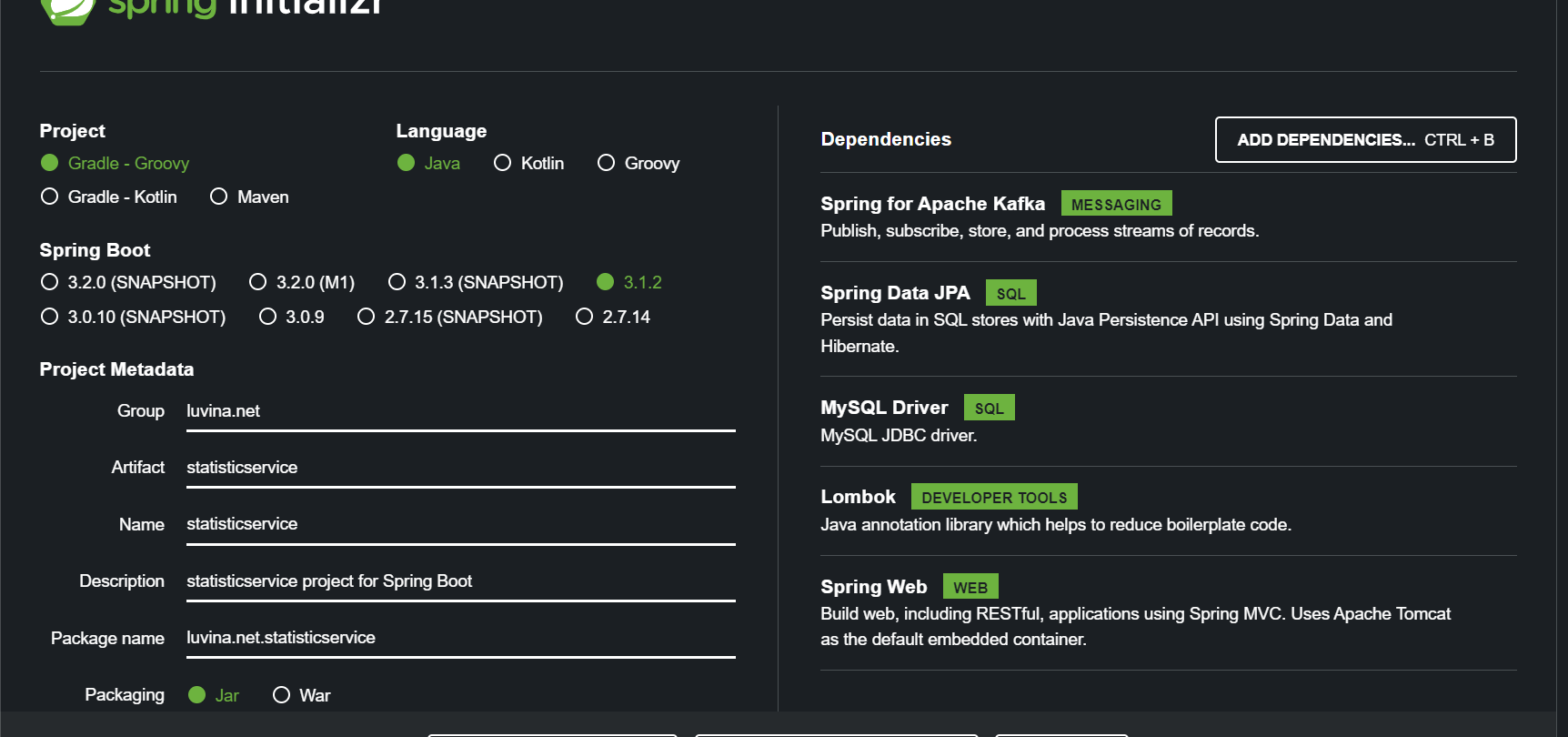


### 2.4. Tạo kafka consumer - statistiscservice

- Tạo statistiscservice với vai trò là 1 consumer

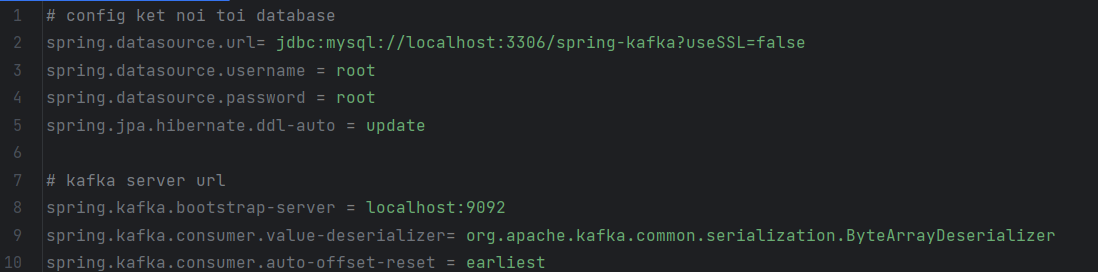
- Tạo project spring boot tại <https://start.spring.io/>

- Chọn các Dependencies cần thiết apache kafka, lombok



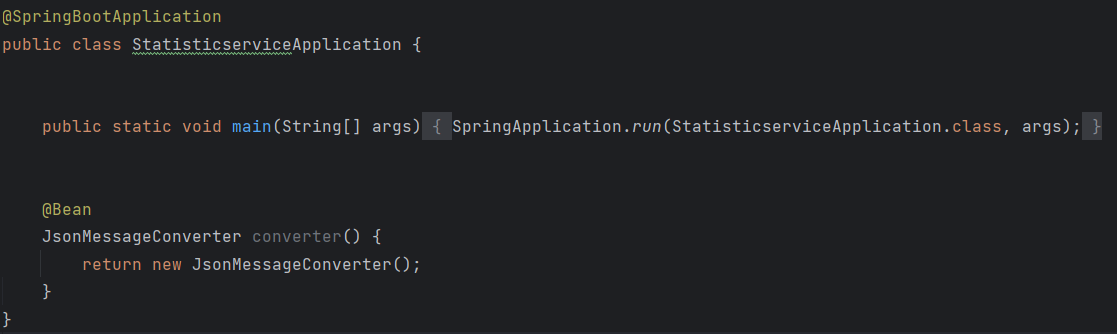
2.4.1. Config

- Config tại file application.properties



2.4.2. StatisticserviceApplication

- Tạo @Bean convert message từ topic thành đối tượng Json



2.4.3. Tạo model statistic

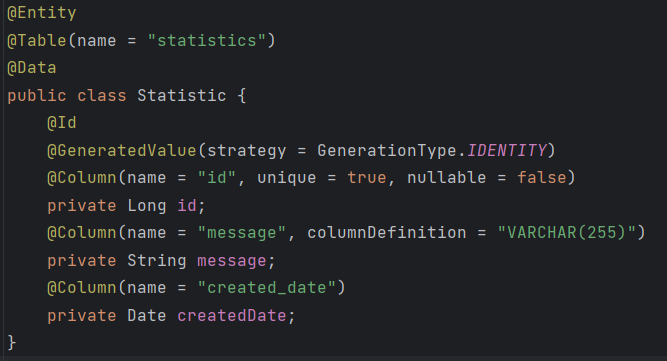
- Tạo model statistic tương ứng với 1 bảng trong database

- Có các thuộc tính giống với model statistic của producer account để có thể tự động convert

+ Sử dụng annotation @Entity để khai báo đây là 1 entity tương ứng với 1 bảng trong database

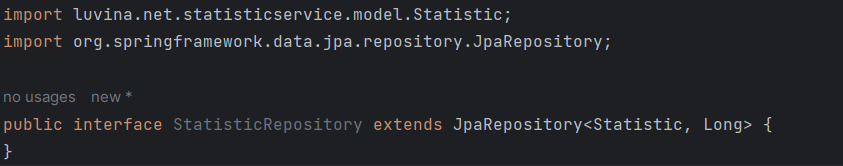
+ @Table: khai báo tên table trong database

+ @Column: Khai báo tên cột trong bảng và các thuộc tính khác



2.4.4. Tạo repository

- Tạo StatisticRepository để thao tác với database thông qua JpaRepository



2.5.5. Tạo service

- Tạo StatisticService để đọc message từ topic và ghi vào database

- @Service: để đánh dấu đây là 1 service

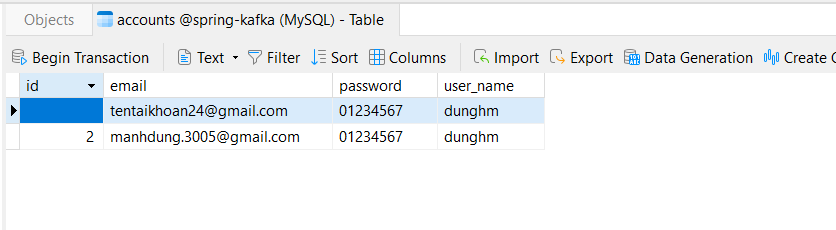
- @Autowired để sử dụng StatisticRepository đã tạo trước đó

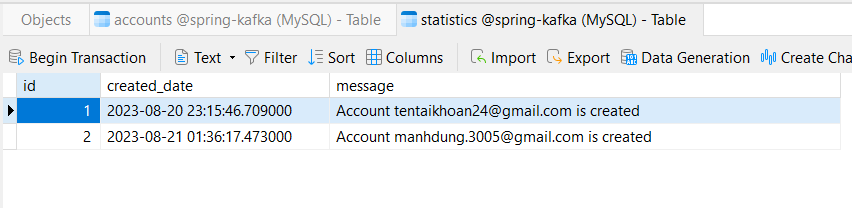
- @KafkaListener: đọc message event từ topic – id: tạo id cho consumer nếu chạy nhiều consumer 1 lúc sẽ có chung 1 id, topics: topic được chỉ định để đọc message



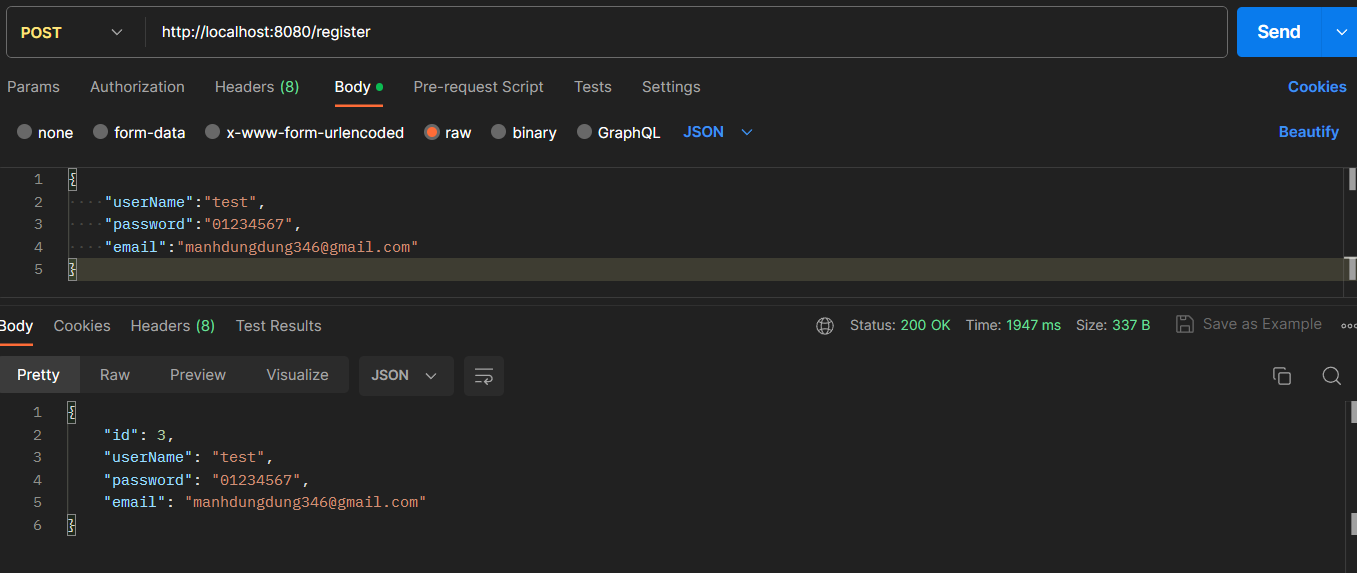
### 3. Chạy chương trình và kiểm tra kết quả

- Database





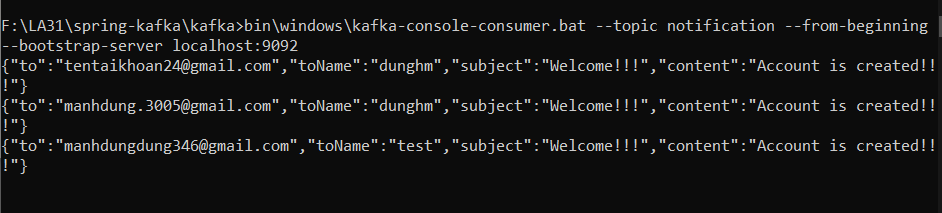
- Gửi request bằng postman



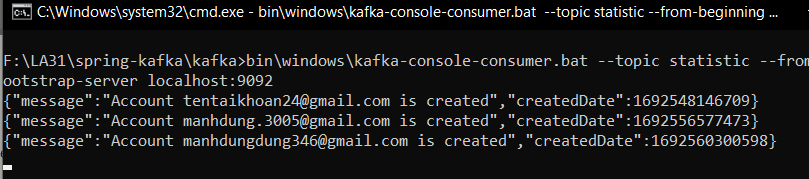
Đã gửi thành công bằng postman

- Kiểm tra message trong các topic

+ Trong topic notification đã có message

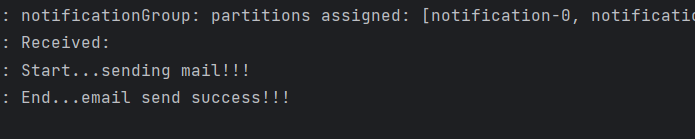


+ Trong topic statistic đã có message

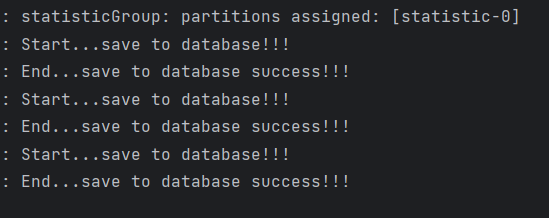


- Kiểm tra log trong termial của các consumer

+ Consumer notification đã hiện thị gửi email thành công

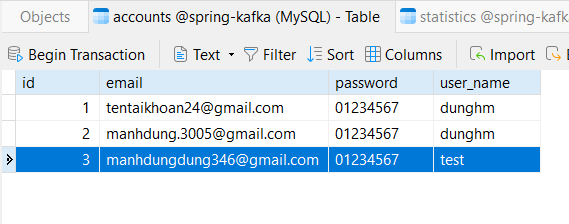


+ Consumer statistic đã thông báo lưu vào database thành công

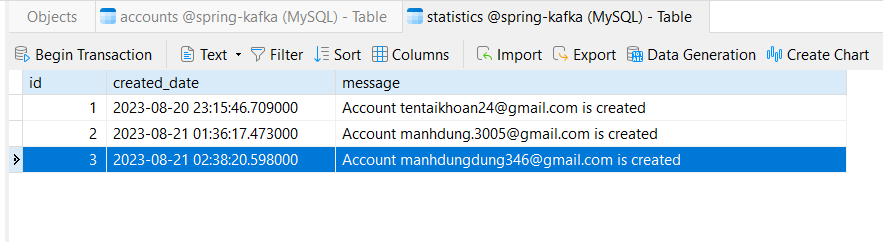


- Kiểm tra trong database

+ Bảng account đã có dữ liệu



+ Bảng statistic đã có dữ liệu



- Kiểm tra email của tài khoản [manhdungdung346@gmail.com](mailto:manhdungdung346@gmail.com)

+ Đã nhận được email đúng theo template

