

# EDA-Demo – Kiến Trúc Event-Driven Đăng Ký Sự Kiện

---

## Mục Tiêu Demo

Trình bày luồng hoạt động của hệ thống microservices áp dụng **Event-Driven Architecture (EDA)**:

- Đăng ký người dùng mới
- Đăng nhập người dùng
- Đăng ký tham gia sự kiện
- Nhận email xác nhận đăng ký sự kiện
- Ghi lại toàn bộ hành vi vào **audit-log**

## Kiến Trúc Tổng Quan

```
[Gateway] → [User Service] → Kafka (USER_CREATED, USER_LOGINED)
                                     ↘
                                     → [Auditlog Service]

[Gateway] → [Registration Service] → Kafka (REGISTRATION_CREATED)
                                     ↘
                                     → [Event Service]
                                     → [Notification Service] → Kafka
(EMAIL_SENT)

[Auditlog Service]
```

Mỗi service độc lập xử lý tác vụ riêng và **giao tiếp thông qua sự kiện Kafka**, thay vì gọi trực tiếp.

---

## Luồng Hoạt Động

### 1. Người dùng đăng ký

- Gateway nhận request → chuyển đến **user-service**
- **user-service** tạo user và phát event **USER\_CREATED**
- **auditlog-service** lắng nghe sự kiện **USER\_CREATED** và ghi log

### 2. Người dùng đăng nhập

- **user-service** xác thực → phát event **USER\_LOGINED**
- **auditlog-service** ghi lại hành vi đăng nhập

### 3. Người dùng đăng ký sự kiện

- Gateway gửi yêu cầu đến `registration-service`
- `registration-service` tạo bản ghi và phát event `REGISTRATION_CREATED`
- Các consumer lắng nghe:
  - `event-service`: cập nhật số lượng người tham gia
  - `notification-service`: gửi email → phát tiếp `EMAIL_SENT`
  - `auditlog-service`: ghi lại hành vi

#### 4. 📧 Email xác nhận

- `notification-service` xử lý event `REGISTRATION_CREATED`
- Gọi `user-service` để lấy email (tự tra cứu – **event-notification**)
- Gửi email → phát `EMAIL_SENT`
- `auditlog-service` ghi lại việc email đã được gửi

### 📖 Mô Hình Sự Kiện (Event Flow)

Event Name	Được phát từ	Ai lắng nghe
<code>USER_CREATED</code>	<code>user-service</code>	<code>auditlog-service</code>
<code>USER_LOGINED</code>	<code>user-service</code>	<code>auditlog-service</code>
<code>REGISTRATION_CREATED</code>	<code>registration-service</code>	<code>event-service</code> , <code>notification-service</code> , <code>auditlog-service</code>
<code>EMAIL_SENT</code>	<code>notification-service</code>	<code>auditlog-service</code>

### 🧠 Kiến Trúc Event-Driven Được Thể Hiện Qua:

Yếu tố	Minh chứng trong hệ thống
<b>Loose coupling</b>	Các service không gọi nhau trực tiếp (trừ khi cần tra cứu)
<b>Event-as-notification</b>	Kafka message chỉ chứa ID, consumer tự tra cứu thêm
<b>Scalable consumers</b>	Có thể thêm consumer mới không ảnh hưởng producer
<b>Audit / Tracking dễ dàng</b>	<code>auditlog-service</code> chỉ cần subscribe Kafka để theo dõi toàn hệ thống

### 🚀 Hướng Dẫn Chạy Demo

📊 Giới thiệu Kafka UI (<http://localhost:8080>)

Kafka UI là công cụ giao diện trực quan giúp theo dõi hoạt động của Kafka, bao gồm:

Thành phần	Mô tả
<b>Clusters</b>	Danh sách các Kafka cluster đang kết nối

Thành phần	Mô tả
<b>Topics</b>	Danh sách các chủ đề Kafka (USER_CREATED, EMAIL_SENT...)
<b>Messages</b>	Xem nội dung message (JSON) được gửi từ producer
<b>Consumer Groups</b>	Xem các consumer đang lắng nghe, vị trí offset, trạng thái
<b>Partitions</b>	Phân vùng của topic, dùng để scale và phân tán

 Bạn có thể click vào từng topic để xem luồng dữ liệu, ai consume, dữ liệu gì đang đi qua Kafka.

## Khởi chạy hệ thống và thao tác thực tế

```
# 1. Khởi động toàn bộ hệ thống
docker-compose up -d --build



# 2. Khởi động frontend
cd frontend
npm install --legacy-peer-deps
npm run dev
```

## Các bước thao tác giao diện người dùng (Frontend + Kafka UI)




Giao diện người dùng được xây bằng Next.js, kết nối qua Gateway. Kafka UI dùng để theo dõi real-time các sự kiện.

- Truy cập giao diện Ứng dụng: <http://localhost:3000>
- Truy cập giao diện Kafka: <http://localhost:8080>



### 1. Đăng ký tài khoản mới

- Chọn nút **Đăng ký**
- Nhập thông tin: tên, email, mật khẩu → bấm **Đăng ký**
-  Kiểm tra trong **Kafka UI** topic **USER\_CREATED** xuất hiện message mới
-  Truy cập **auditlog-service** hoặc **pgadmin** để xem log tạo user

### 2. Đăng nhập

- Chọn nút **Đăng nhập**
- Nhập email + mật khẩu → bấm **Đăng nhập**
-  Kiểm tra topic Kafka **USER\_LOGINED**
-  Xem log đăng nhập trong **auditlog**
-  Nhận token Bearer (lưu vào LocalStorage hoặc DevTool để dùng cho bước tiếp theo)

### 3. Đăng ký tham gia sự kiện

- Chọn sự kiện và bấm **Đăng ký**
-  Kafka emit: **REGISTRATION\_CREATED**
-  Xem trong Kafka UI các topic:

- `event-service` xử lý cập nhật số lượng
- `notification-service` gửi email → topic `EMAIL_SENT`
- `auditlog-service` ghi nhận tất cả các hành vi

#### 4. Xác nhận email được gửi

- Console log từ `notification-service` in ra email đã gửi
- Kafka UI hiển thị message ở topic `EMAIL_SENT`
- DB `auditlog` lưu bản ghi email
- Có thể vào email để kiểm tra

#### 5. Quan sát toàn bộ hệ thống qua Kafka UI

- Truy cập: `http://localhost:8080`
- Chọn các topic để theo dõi:
  - `USER_CREATED`
  - `USER_LOGINED`
  - `REGISTRATION_CREATED`
  - `EMAIL_SENT`
- Theo dõi thời gian emit, nội dung message, consumer group...

```
json {  
  "name": "Alice",  
  "email": "[alice@example.com](mailto\:alice@example.com)",  
  "password": "123456"  
}
```

- ☒ Kiểm tra log Kafka: có `USER_CREATED`
- ☒ Truy cập `auditlog-service` DB: thấy bản ghi đăng ký

---

### Các Service Tham Gia

Service	Chức năng chính
<code>user-service</code>	Đăng ký / đăng nhập / phát event
<code>registration-service</code>	Xử lý đăng ký sự kiện
<code>event-service</code>	Cập nhật dữ liệu sự kiện khi có đăng ký
<code>notification-service</code>	Gửi email và phát event <code>EMAIL_SENT</code>
<code>auditlog-service</code>	Lắng nghe tất cả sự kiện và ghi log
<code>gateway</code>	Tiếp nhận request từ client

---

### Thông Tin Kafka Topics & Consumer Groups

#### Danh sách Topics đang sử dụng

Topic Name	Ý nghĩa
<code>user.created</code>	Phát khi người dùng mới được tạo
<code>user.logged_in</code>	Phát khi người dùng đăng nhập thành công
<code>user.updated</code>	(Dự phòng) Phát khi thông tin user được cập nhật
<code>event.created</code>	(Dự phòng) Phát khi có event mới
<code>event.updated</code>	Phát khi event được cập nhật
<code>registration.created</code>	Phát khi có người đăng ký sự kiện
<code>registration.cancelled</code>	(Dự phòng) Phát khi người dùng hủy đăng ký
<code>notification.sent</code>	Ghi nhận đã gửi notification (email)
<code>notification.failed</code>	(Dự phòng) Ghi nhận lỗi khi gửi notification
<code>audit.logged</code>	Ghi nhận các hành vi thành công
<code>audit.failed</code>	(Dự phòng) Ghi nhận hành vi thất bại
<code>__consumer_offsets</code>	(Hệ thống) Kafka sử dụng để theo dõi offset các consumer group

## Danh sách Consumer Groups tiêu biểu

Consumer Group	Mô tả
<code>audit-user-created</code>	Ghi log sự kiện tạo user
<code>audit-user-logged</code>	Ghi log hành vi đăng nhập
<code>audit-registration-created</code>	Ghi log hành vi đăng ký sự kiện
<code>audit-event-updated</code>	Ghi log khi event cập nhật
<code>notification-group</code>	Lắng nghe REGISTRATION_CREATED để gửi email
<code>event-group</code>	Lắng nghe REGISTRATION_CREATED để cập nhật sự kiện
<code>audit-audit-logged</code>	Lưu trữ toàn bộ log gửi qua topic <code>audit.logged</code>
<code>audit-email-sent</code>	Ghi log việc gửi email thành công

 Các group như `audit-*` có thể gom vào 1 group lớn nếu không cần tách biệt chi tiết.

## Tổng Kết

Hệ thống minh họa rõ:

- Mô hình **event-driven** với Kafka
- Sử dụng **event-notification pattern** (tra cứu dữ liệu khi cần)
- **Khả năng mở rộng** bằng cách thêm consumer
- **Tách biệt trách nhiệm rõ ràng** giữa các service

## Kịch bản demo chi tiết (8–12 phút)

- **Mục tiêu:** Trình diễn kiến trúc Event-Driven, cho thấy flow đăng ký user → đăng nhập → đăng ký tham gia sự kiện → gửi email → ghi log/audit, đồng thời quan sát message trên Kafka UI.

### 1) Chuẩn bị và khởi động dịch vụ

- **Yêu cầu:** Docker + Docker Compose, Node 18+.
- **Khởi động backend + hạ tầng:**

```
cd /Users/Kiet/Documents/School/Junior/SW_architecture/LAB_03/Event-Driven-Demo
docker-compose up -d --build
```

- **Khởi động frontend:**

```
cd /Users/Kiet/Documents/School/Junior/SW_architecture/LAB_03/Event-Driven-Demo/frontend
npm install --legacy-peer-deps
# (tuỳ chọn) đảm bảo frontend trở gateway đúng (mặc định đã là 3007)
export NEXT_PUBLIC_GATEWAY_URL=http://localhost:3007
npm run dev
```

- **Mở giao diện:**
  - Ứng dụng: <http://localhost:3000>
  - Kafka UI: <http://localhost:8080>

Lưu ý: Email mặc định đã set là [truongkiet771@gmail.com](mailto:truongkiet771@gmail.com) trong `docker-compose.yml` và `notification-service/.env`. Nếu trước đó `notification-service` đang chạy, hãy restart:

```
docker-compose up -d --build notification-service
```

### 2) Giới thiệu nhanh kiến trúc (30–45s)

- Frontend gọi **gateway** (Fastify), gateway publish/route sang các service.
- Các service giao tiếp qua Kafka: topic dạng dot-case như `user.created`, `registration.created`, `notification.sent`, `audit.logged...`
- Quan sát Kafka UI: Topics đã được tạo sẵn (script `shared/init-kafka-topics.sh`).

### 3) Demo 1 – Đăng ký người dùng (1–2 phút)

- Trên UI, vào trang đăng ký user, nhập: username, email, password → bấm Đăng ký.
- Kỳ vọng:
  - UI: thông báo đăng ký thành công.

- Kafka UI:
  - Topic: `user.created` có message mới (payload chứa `userId/username/email`).
  - Topic: `audit.logged` có log tương ứng hành vi tạo user.

#### 4) Demo 2 – Đăng nhập (1 phút)

- Trên UI, đăng nhập bằng email + password vừa tạo.
- Kỳ vọng:
  - UI: login OK, frontend lưu `userId` và `username` vào LocalStorage.
  - Kafka UI:
    - Topic: `user.logged_in` có message mới.
    - Topic: `audit.logged` ghi lại hành vi login.

#### 5) Demo 3 – Đăng ký tham gia sự kiện (2–3 phút)

- Trên UI, vào danh sách sự kiện (được seed sẵn), chọn 1 event → bấm “Đăng ký”.
- Kỳ vọng:
  - UI: thấy số lượng người tham gia event tăng (do `event-service` consume `registration.created` và cập nhật).
  - Kafka UI:
    - Topic: `registration.created` có message mới (chứa `userId`, `eventId`).
    - Topic: (tùy logic) có thể thấy `event.updated` nếu `event-service` phát sự kiện cập nhật.
    - Topic: `notification.sent` xuất hiện sau khi `notification-service` gửi email.
    - Topic: `audit.logged` ghi nhận đầy đủ các hành vi trên.
- Email:
  - Kiểm tra hộp thư `truongkiet771@gmail.com` để thấy email xác nhận.
  - Có thể mở logs container `notification-service` để thấy log “email sent”.

#### 6) Điểm nhấn “Event-as-notification” (30–45s)

- Nhấn mạnh message chỉ mang ID; consumer tự tra cứu dữ liệu khi cần (ví dụ: `notification-service` tra email user khi gửi).

#### 7) Quan sát tổng thể trên Kafka UI (1 phút)

- Mở từng topic và nói nhanh về luồng dữ liệu:
  - `user.created`, `user.logged_in`
  - `registration.created`
  - `notification.sent`
  - `audit.logged`
- Cho thấy consumer groups hoạt động, offset di chuyển.

#### 8) Thu dọn (tùy chọn)

```
docker-compose down -v
```

## Ghi nhớ nhanh khi thuyết trình

- **Địa chỉ:** UI <http://localhost:3000>, Gateway <http://localhost:3007>, Kafka UI <http://localhost:8080>
- **Topics chính:** [user.created](#), [user.logged\\_in](#), [registration.created](#), [notification.sent](#), [audit.logged](#)
- **Email demo:** gửi đến [truongkiet771@gmail.com](mailto:truongkiet771@gmail.com)