Khoa Mạng máy tính và Truyền thông - UIT

6

CÔNG NGHỆ
INTERNET OF THINGS
HIỆN ĐẠI

Hoàn thiện giải pháp IoT

Lưu hành nội bộ

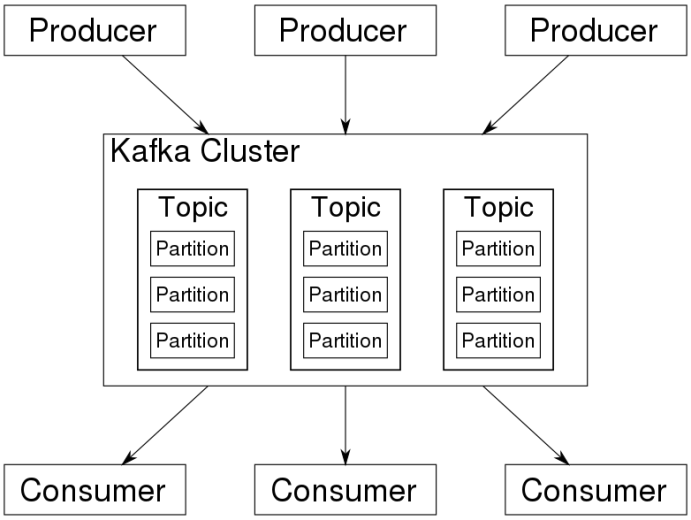
A. MỤC TIÊU

- Tổng hợp tất cả các bài thực hành ở các tuần trước.
- Hoàn thiện mô hình IoT có đầy đủ các thành phần.
- Tìm hiểu về Apache Kafka và thực hiện kịch bản đơn giản.
- Giúp sinh viên có thể tự phát triển một giải pháp IoT một cách hoàn thiện.

B. GIỚI THIỆU

1. Apache Kafka là gì?

Apache Kafka là một nền tảng theo kiến trúc phân tán cho phép lưu trữ sự kiện và xử lý dữ liệu luồng mã nguồn mở được phát triển bởi Apache Software Foundation được viết bằng Java và Scala.



Hình 1. Mô hình tổng quan Apache Kafka

Kafka là một trong những nền tảng message publish / subscribe phân tán mã nguồn mở phổ biến nhất hiện nay và được xây dựng với mục đích để xử lý dữ liệu streaming real-time.

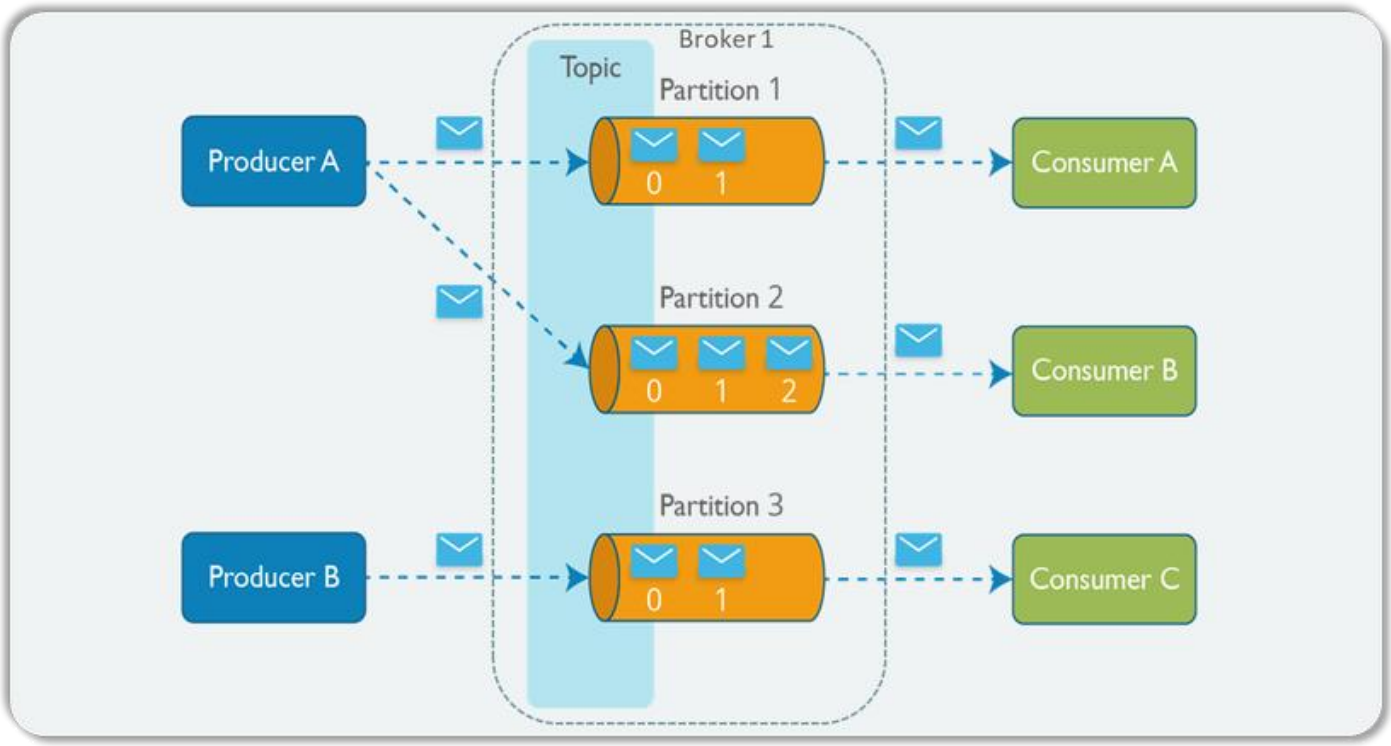
2. Tại sao phải sử dụng Apache Kafka?

- Kafka cho phép xử lý dữ liệu dạng "Batch" và thời gian thực một cách hiệu quả.

- Kafka có khả năng mở rộng linh hoạt và có thể xử lý hàng triệu thông điệp mỗi giây. Điều này làm cho nó phù hợp cho các hệ thống với yêu cầu lưu trữ và xử lý lớn.
- Dữ liệu trong Kafka được lưu trữ theo cách an toàn và bền vững.
- Kafka hỗ trợ xử lý dòng thông điệp (stream processing) trực tiếp trên dữ liệu trong hệ thống, cho phép người phát triển xây dựng các ứng dụng xử lý dữ liệu theo thời gian thực.

3. Các thành phần chính

- **Producer**: là thành phần chịu trách nhiệm cho việc gửi thông điệp vào Kafka Topic. Producer gửi thông điệp theo chủ đề (topic), mà sau đó sẽ được chia sẻ và xử lý bởi các Consumer.



Hình 2. Mô các thành phần của Apache Kafka [1]

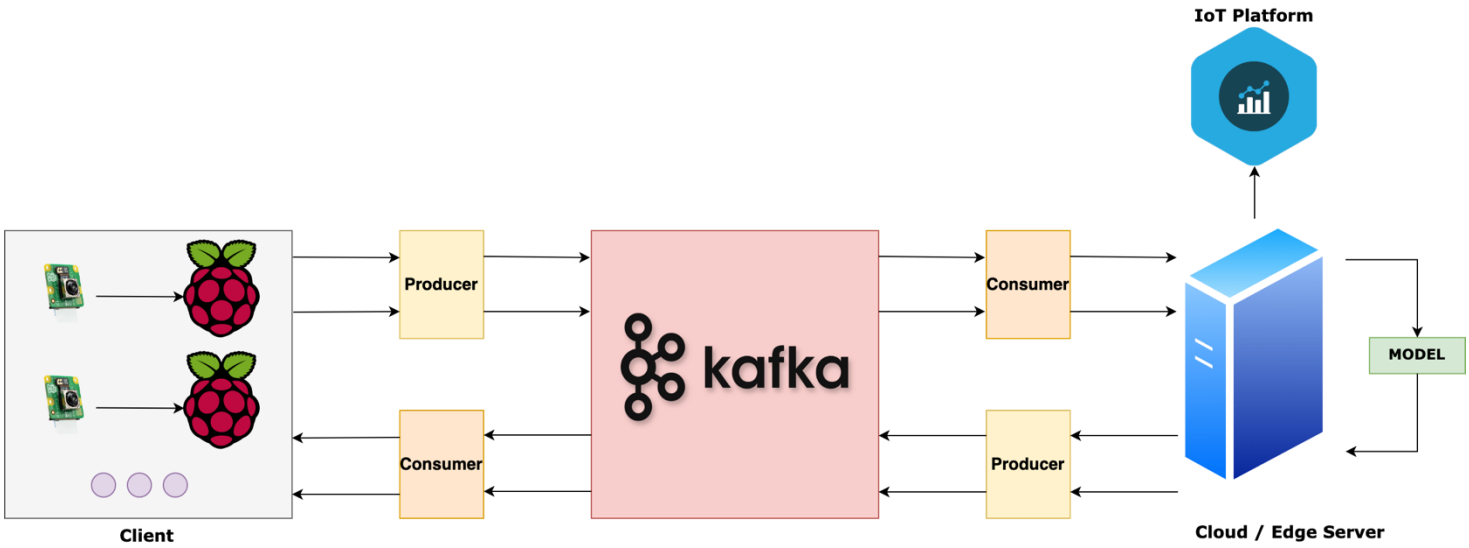
[1] https://topdev.vn/blog/kafka-la-gi/

- **Consumer**: là thành phần chịu trách nhiệm đọc và xử lý thông điệp từ Kafka Topic. Một Topic có thể có nhiều Consumer, mỗi Consumer đọc thông điệp từ một phần của Topic.
- **Broker**: Kafka sử dụng khái niệm Broker để đại diện cho các máy chủ trong môi trường Kafka. Mỗi Broker là một máy chủ Kafka có khả năng xử lý và lưu trữ thông điệp.
- **Topic**: là một loại kênh hoặc danh mục nơi thông điệp được phân loại. Producer gửi thông điệp vào một Topic, và các Consumer đọc thông điệp từ một hoặc nhiều Topic.
- **Partition**: Mỗi Topic có thể được chia thành nhiều phần (partition), giúp tăng khả năng mở rộng và xử lý đồng thời.
- **ZooKeeper**: là một dịch vụ quản lý Kafka Cluster, giúp theo dõi và quản lý các thành phần trong Kafka Cluster.

C. THỰC HÀNH

1. Mô hình

Tại bài thực hành này, sinh viên tiến hành xây dựng một hệ thống điểm danh bằng khuôn mặt có thể phục vụ được nhiều người dùng, được mô tả như hình dưới đây:



Hình 3. Mô hình tổng quan bài thực hành

Hệ thống bao gồm các thành phần:

- **Client:** bao gồm các IoT Devices như Raspberry Pi 3, Raspberry Pi Camera Module V2 (tương tự như bài thực hành số 5).
- **Edge / Cloud server:** PC / Laptop.
- **Model:** mô hình nhận diện khuôn mặt FaceNet (tương tự như bài thực hành số 5).
- **IoT Flatform:** giao diện ứng dụng IoT cơ bản (tương tự bài thực hành số 4).

2. Chạy thử nghiệm sử dụng Kafka

Mô tả: Sinh viên sử dụng máy tính cá nhân để kiểm tra và sử dụng Kafka với mã nguồn được cung cấp sẵn. Quá trình được thực hiện trên Docker.

- **Bước 1:** Sinh viên có thể tải về tham khảo sử dụng Kafka được lưu trữ tại đây.
- **Bước 2:** Cài đặt các thư viện.
- **Bước 3:** Cấu hình địa chỉ IP trong các file: config.py, docker-compose.yml
- **Bước 4:** Cài đặt Kafka và Zookeeper:

```
docker-compose -f docker-compose.yml up -d
```

- **Bước 5:** Kết nối Kafka shell:

```
docker exec -it  kafka bin/bash
cd opt/bitnami/kafka/bin
```

- **Bước 6:** Kiểm tra danh sách các Topic đã tạo:

```
kafka-topics.sh --list --bootstrap-server 192.168.100.171:9092
```

- **Bước 7:** Kiểm tra gửi và nhận dữ liệu: thực thi 2 file dưới đây, mặc định Topic TEST sẽ được tạo tự động:

```
python test_send.py
python test_receive.py
```

- **Bước 8:** Gửi và nhận dữ liệu: thực thi 2 file dưới đây, mặc định Topic send_image và receive_result sẽ được tạo tự động:

```
python recognize.py
python send_image.py
```

- **Bước 9:** Một số câu lệnh với Kafka:

```
kafka-topics.sh --create --bootstrap-server
192.168.100.171:9092 --replication-factor 1 --partitions 1
--topic send_image
kafka-topics.sh --create --bootstrap-server
192.168.100.171:9092 --replication-factor 1 --partitions 1
--topic receive_result
```

D. YÊU CẦU VÀ NỘP BÀI

1. Yêu cầu

1. Thu thập hình ảnh các thành viên trong nhóm, sau đó huấn luyện lại mô hình Facenet với độ chính xác trên 60%. (Sử dụng mã nguồn cung cấp trong bài thực hành số 5).

2. Sử dụng Apache Kafka làm kênh giao tiếp giữa Client và Edge / Cloud Server. Cụ thể, Client sử Kafka Producer để gửi hình ảnh tới Topic trên Kafka Broker. Lúc đó, Edge / Cloud Server sử dụng Kafka Consumer để nhận hình ảnh đó. Hình ảnh đưa vào mô hình Face Net để tiến hành nhận diện. Kết quả được Edge / Cloud Server trả về Client cũng thông qua quá trình pub/sub tương tự như trên. Client nhận kết quả và xuất ra màn hình.

3. Kết quả nhận diện đồng thời được gửi lên IoT Flatform để lưu trữ quản lý. Yêu cầu lưu trữ các thông tin như: Timestamp (h:m:s d/m/y), Room ID, Student Name.

2. Nộp bài

- Sinh viên tìm hiểu và thực hành theo hướng dẫn. Thực hiện **nhóm.**
- Sinh viên báo cáo kết quả thực hiện và nộp bài bằng file. Trong đó:

- Trình bày chi tiết quá trình thực hành và trả lời các câu hỏi nếu có (kèm theo các ảnh chụp màn hình tương ứng).
- Giải thích các kết quả đạt được.
- Tải mẫu báo cáo thực hành và trình bày theo mẫu được cung cấp.
- Nộp lại các file liên quan tại github.

Nén tất cả các file và đặt tên file theo định dạng theo mẫu: **NhomY-LabX_MSSV1_MSSV2** Ví dụ: Nhom1-Lab06_25520001_25520002

- Nộp báo cáo trên theo thời gian đã thống nhất tại website môn học.
- Các bài nộp không tuân theo yêu cầu sẽ KHÔNG được chấm điểm.

HẾT

Chúc mừng các bạn đã hoàn thành khóa học!