BÁO CÁO TÌM HIỂU DỊCH VỤ TRUYỀN THÔNG ĐIỆP: RABBITMQ

Họ và tên: Dương Công Minh
Mã số sinh viên: 22010009
Môn học: Ứng dụng phân tán

1. Mở đầu

Mục tiêu báo cáo này là giới thiệu cơ chế hoạt động, các chức năng chính, và hướng dẫn cài đặt cơ bản cho RabbitMQ — một message broker phổ biến sử dụng AMQP (Advanced Message Queuing Protocol).

2. Giới thiệu

RabbitMQ là một message broker mã nguồn mở (open-source) được viết bằng Erlang. Nó hỗ trợ mô hình truyền thông bất đồng bộ giữa các thành phần (producer — broker — consumer), cho phép tách rời các dịch vụ, cân bằng tải, và đảm bảo độ tin cậy của truyền tin trong hệ phân tán.

3. Cơ chế hoạt động chính

3.1. AMQP và mô hình tổng quát

RabbitMQ triển khai AMQP 0-9-1 (phiên bản phổ biến). Trong mô hình này, producer không gửi trực tiếp đến queue mà gửi tới một exchange; exchange chịu trách nhiệm định tuyến (routing) đến một hoặc nhiều queue theo các binding.

3.2. Thành phần chính

- Producer: gửi message tới exchange.
- Exchange: nơi nhận message và định tuyến sang queue theo quy tắc (binding). Các loại exchange phổ biến: direct, fanout, topic, headers.
- Queue: nơi lưu message chờ consumer xử lý.
- Binding: liên kết giữa exchange và queue, có thể kèm routing key.
- Consumer: nhận và xử lý message từ queue.
- Virtual Hosts (vhost): phân vùng logic cho các ứng dụng; giúp phân quyền và cô lập tài nguyên.

3.3. Dòng đời của một message

1. Producer publish message tới một exchange kèm routing key.
2. Exchange xác định queue đích dựa vào binding và routing key.
3. Message được đưa vào queue; nếu queue durable và message persistent thì lưu trên đĩa.
4. Consumer fetch message (push hoặc pull), xử lý và gửi acknowledgement (ack) để broker biết có thể xóa message.
5. Nếu consumer không ack, message có thể được re-queue hoặc dead-lettered tùy cấu hình.

3.4. Đảm bảo chất lượng dịch vụ

- Durable queues và persistent messages để chống mất mát khi broker khởi động lại.
- Acknowledgements (ACK/NACK) để đảm bảo xử lý lại khi consumer thất bại.
- Dead Letter Exchanges (DLX) để xử lý message lỗi/hết hạn.

4. Chức năng nổi bật

- Hỗ trợ nhiều mô hình messaging: point-to-point, publish/subscribe, routing theo pattern (topic).
- Hỗ trợ plugin: management UI, shovel, federation, delayed message exchange, prometheus exporter,…
- Hỗ trợ clustering và replication (quorum queues) để tăng khả năng chịu lỗi và mở rộng.
- Hỗ trợ nhiều ngôn ngữ qua client libraries (pika for Python, amqp libraries for Java/.NET/Node.js,…).
- Quản trị và giám sát qua Management Plugin (HTTP API, UI, Prometheus metrics via plugin).

5. Thiết lập clustering và tính sẵn sàng cao

- Có hai mô hình chính: cluster (nhiều node chia sẻ metadata) và replication/queue mirroring.
- RabbitMQ khuyến nghị dùng quorum queues cho workloads yêu cầu an toàn dữ liệu hơn mirror queues cũ.
- Các bước cơ bản để thành lập cluster: cài RabbitMQ trên mỗi node, đảm bảo các node có thể kết nối nhau, chọn một seed node và dùng rabbitmqctl join_cluster trên các node khác. Sau đó khởi động lại dịch vụ.

6. Ưu/nhược điểm

Ưu điểm:

- Triển khai nhanh, nhiều plugin, cộng đồng và tài liệu lớn.
- Hỗ trợ patterns messaging phong phú (routing, topic, fanout).
- Management UI thân thiện cho quản trị.

Nhược điểm:

- Với workloads throughput cực lớn (huỷ/streaming), Kafka thường ưu thế hơn.
- Cần quản lý phiên bản Erlang tương thích; đôi khi gây ra vấn đề khi nâng cấp.

7. Các trường hợp sử dụng điển hình

- Task queues, background job processing (ví dụ Celery với RabbitMQ).
- Routing messages giữa microservices khi cần patterns phức tạp.
- Hệ thống cần ack/retry, dead-lettering, hoặc delayed processing.

8. Kết luận

RabbitMQ là lựa chọn tốt cho nhiều ứng dụng microservices và task queue khi cần mô hình messaging phong phú, độ tin cậy và quản trị dễ dàng. Đối với hệ thống cần throughput cực lớn hoặc retention lâu dài, cân nhắc Kafka.