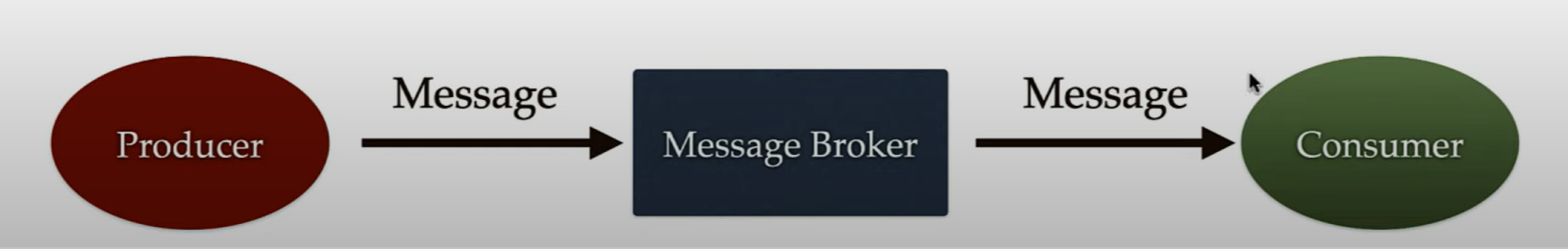
**MESSAGE QUEUE**

1. **Message Queue là gì?**

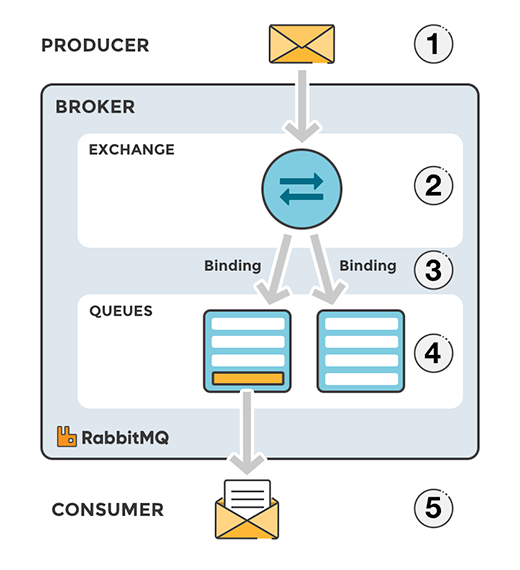
* Message queuing allows applications to communicate by sending messages to each other. The message queue provides tempory messages storage when the destination program is busy or not connected.
* A message queue is made up of a producer, a broker (the message queue software), and a consumer.
* A Message Queue is a form of asynchronous, decoupled communication between software components (process, thread or service), enabling reliable message delivery without requiring sender and receiver to interact simultaneously.
* RabbitMQ originally implements the **A**dvanced **M**essage **Q**ueuing **P**rotocol (AMQP). But now RabbitMQ also supports serveral other API protocols such as STOMP, MQTT and HTTP.



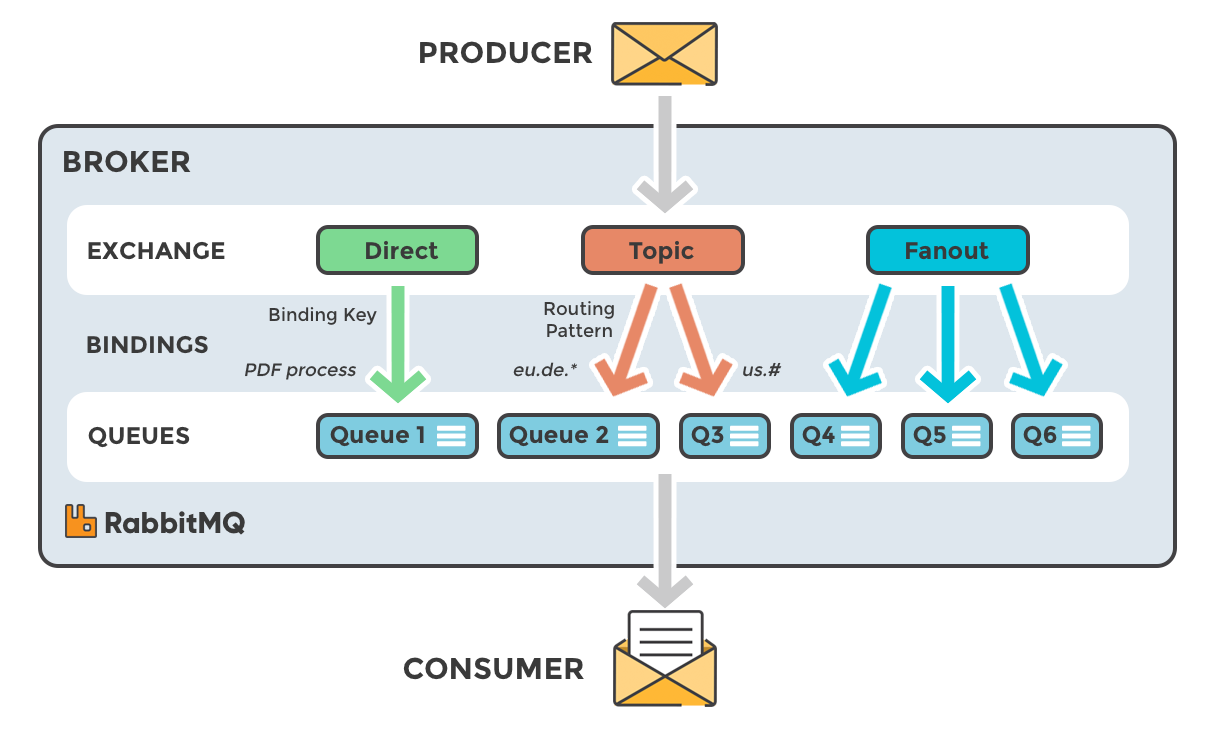
1. **RabbitMQ hoạt động như nào?**

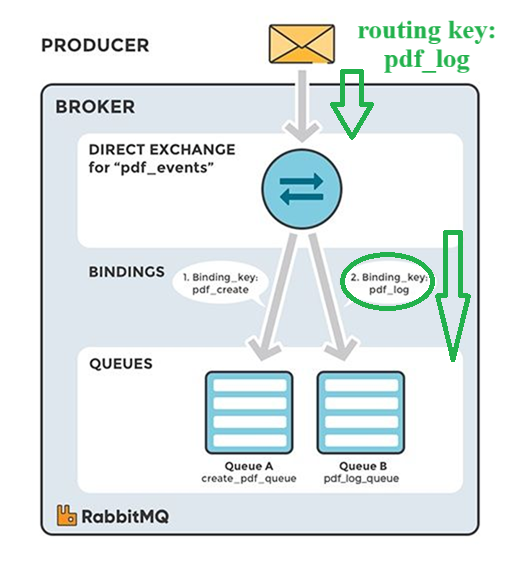
Luồng message trong RabbitMQ hoạt động như sau:

* **Producer** gửi một message tới một exchange. Exchange phải được chỉ định type khi được tạo.
* **Exchange** nhận được message và chịu trách nhiệm định tuyến message. Việc trao đổi sẽ tính đến các thuộc tính của message và loại exchange, chẳng hạn như routing key.
* Giữa exchange và queue phải thiết lập bindings từ trước. **Exchange** định tuyến message vào queue tùy thuộc vào thuộc tính của message. Trong ví dụ trên, có 2 bindings từ exchange với 2 queue khác nhau.
* Các **message** vẫn ở trong queue cho đến khi chúng được consumer xử lý.
* **Consumer** xử lý message.

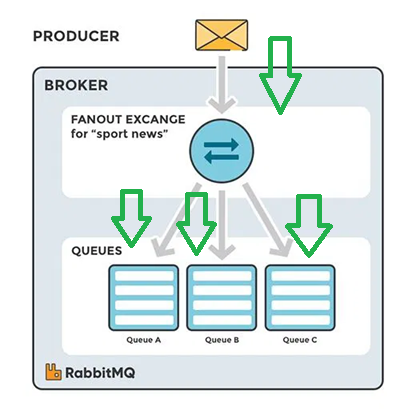


1. **Các thành phần trong RabbitMQ?**

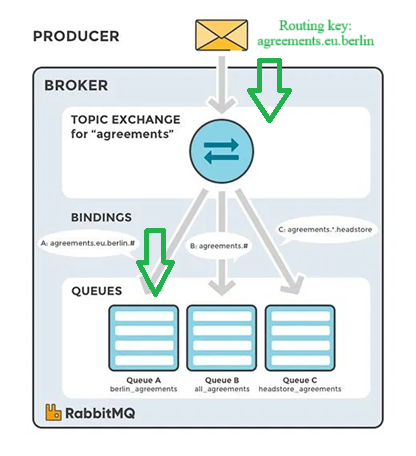
* **Producer** is an application that sends messages to the RabbitMQ broker.
* **Message** (tin nhắn) là đơn vị cơ bản để trao đổi thông tin trong RabbitMQ. Nó bao gồm:
  + **Message Body** (payload): chứa dữ liệu thực tế cần được gửi, có thể ở nhiều định dạng khác nhau như JSON, XML, binary hoặc bất kỳ cấu trúc nào.
  + **Message Properties** (thuộc tính): bao gồm các atribute và metadata khác nhau đi kèm với message. Dưới đây là danh sách một số property điển hình:
    - Routing key (Khoá định tuyến): được exchange dùng để xác định cách định tuyến message tới queue (phụ thuộc vào exchange type).
    - Message ID: mã định danh duy nhất xác định message.
    - Timestamp: thời gian khi message được tạo.
    - Headers (Tiêu đề): Tiêu đề tuỳ chỉnh chứa thông tin bổ sung về message.
    - Delivery Mode (Chế độ chuyển phát): chỉ định chế độ chuyển phát, liệu có nên lưu message vào đĩa để duy trì độ bền hay không.
    - Expiration (Hạn dùng): thời gian message bị loại bỏ nếu không được sử dụng.
    - Priority (Độ ưu tiên): mức độ ưu tiên của message.
    - Content type (Loại nội dung): mô tả định dạng của nội dung message (ví dụ: application/json).
    - Content encoding (Mã hoá nội dung): chỉ định kiêủ mã hoá được sử dụng cho nội dung message (ví dụ: utf-8).
* **Bindings** (ràng buộc): là các ràng buộc được thiết lập để xác định mối quan hệ giữa exchange và queue. Chúng chỉ định các quy tắc định tuyến để gửi message từ exchange tới queue thích hợp. Một queue có thể liên kết với nhiều exchange và một exchange có thể có nhiều liên kết tới các queue khác nhau.
* **Exchange** (sàn giao dịch) đóng vai trò như một bộ định tuyến message. Khi producer gửi message tới RabbitMQ, message sẽ được gửi tới exchange. Sau đó, exchange sẽ định tuyến message đến queue thích hợp.
* 
* Có 5 loại exchange:
  + **Direct exchange**: message được định tuyến đến queue dựa trên routing key. Nếu routing key của message khớp với routing key của queue, message sẽ được chuyển đến queue đó. Ngược lại, nếu routing key của message không khớp với bất kỳ routing key của queue nào, message sẽ bị loại bỏ.



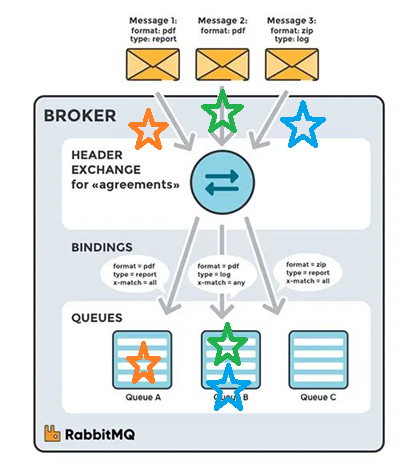
* + **Default exchange**: là một direct exchange mặc định, được khai báo trước mà không có tên, cho phép định tuyến message trực tiếp tới queue với tên là routing key của message. Mỗi queue đều tự động liên kết với exchange mặc định bằng một routing key giống tên của queue.
  + **Fanout exchange**: message được định tuyến đến tất cả các queue được liên kết với nó, bất kể chúng có routing key như nào.



* + Topic exchange: message được định tuyến đến queue dựa trên topic, cho phép sử dụng regular expression pattern trong routing key như \* để đại diện cho một ký tự và # đại diện cho không hoặc nhiều ký tự.



* + **Header exchange**: message được định tuyến dựa trên thuộc tính header của message thay vì routing key. Tham số x-match trong binding xác định cách so sánh các cặp key/value của header trong message với header của binding. Có 2 giá trị cho tham số x-match là any và all.
    - x-match: any: chỉ cần một thuộc tính trong header của message phải khớp với header của binding.
    - x-match: all: tất cả các thuộc tính trong header của message phải khớp với header của binding.



* **Queues**: queue là một danh sách các message được lưu trữ cho tới khi chúng được tiêu thụ bởi consumer.
* **Consumer** is an application that reads messages from the RabbitMQ broker.
  + **Chế độ hoạt động:** Khi một message mới đến queue, consumer sẽ tiêu thụ message thông qua 2 cơ chế hoạt động là:
* **Cơ chế push (Basic.Consume):** RabbitMQ push (đẩy) message tới consumer bất cứ khi nào có message mới trong queue. Nhờ vậy, client nhận được dữ liệu real-time hoặc gần real-time với độ trễ thấp và phản ứng với thay đổi một cách tức thì.

🡪 Do RabbitMQ là một message broker có mục đích chính là đảm bảo rằng message được gửi đến consumer nhanh chóng và hiệu quả nhất có thể, nên push trở thành chế độ hoạt động mặc định trong RabbitMQ.

* **Cơ chế pull (Basic.Get):** consumer chủ động poll message từ queue. Consumer gửi một yêu cầu Basic.Get tới RabbitMQ khi nó trong trạng thái sẵn sàng xử lý message. RabbitMQ sẽ gửi lại message từ queue nếu được.

🡪 Chế độ pull thích hợp dùng cho các trường hợp consumer muốn truy xuất message theo yêu cầu thay vì liên tục được push message.

* + **Delivery acknowledgment mode** (Chế độ xác nhận chuyển phát):chọn giữa 2 chế độ xác nhận chuyển phát message:
    - Automatic: không yêu cầu xác nhận của consumer, còn gọi là "fire and forget"
    - Manual: yêu cầu xác nhận của consumer

"Fire and forget" là một thuật ngữ thường được sử dụng trong lĩnh vực công nghệ thông tin và truyền thông để mô tả một hành động hoặc một phương thức giao tiếp mà người gửi thông điệp hoặc yêu cầu không chờ đợi một phản hồi, xác nhận hoặc kết quả từ người nhận.

1. **Tại sao lại cần Message Queue/Ưu điểm của Message Queue?**

* Here are the core points:
  + **Asynchronous Communication:** Producers send messages to the queue and continue processing immediately. Consumers retrieve and process messages later, at their own pace. This decouples sender and receiver lifecycles and smooths out spikes in load.
  + **Decoupling of Components:** The sender (producer) and receiver (consumer) only need to agree on the queue name and message format—they don’t need to know about each other’s implementation, uptime, or location.
  + **Reliability & Durability:** Queues are managed by a broker (e.g., RabbitMQ, Apache Kafka, Amazon SQS) which can persist messages to disk, replicate them across nodes, and ensure **at‑least‑once** or **exactly‑once** delivery semantics, even in the face of failures.
  + **Load Leveling:** By buffering messages, a queue can absorb bursts of incoming work, allowing consumers to process at a steady rate without being overwhelmed. This improves overall system resilience and responsiveness.
  + **Open source, Lightweight.**
  + **Message Prioritization**: RabbitMQ cho phép các message được chỉ định mức độ ưu tiên khác nhau. Điều này đảm bảo rằng các message có mức độ ưu tiên cao sẽ được xử lý trước các message có mức độ ưu tiên thấp hơn.
  + **Layers of security** (Nhiều lớp bảo mật): Trong RabbitMQ có nhiều lớp bảo mật. Kết nối của client có thể được bảo mật bằng cách thực thi giao tiếp chỉ có SSL và xác thực chứng chỉ client. Quyền truy cập của người dùng có thể được quản lý ở máy chủ ảo, cung cấp sự cô lập mức độ cao cho message và tài nguyên. Ngoài ra, quyền truy cập cấu hình, đọc từ queues (hàng đợi) và ghi vào exchanges (sàn giao dịch) được quản lý bằng regex.

1. **Nhược điểm của Message Queue**

* **Learning curve** (Đường cong học tập): Việc thiết lập và định cấu hình RabbitMQ có thể phức tạp, đặc biệt đối với những người mới làm quen với khái niệm message queuing. Quản lý phù hợp đòi hỏi sự hiểu biết về các khái niệm cốt lõi trong RabbitMQ như queues, exchanges, bindings (ràng buộc), ...
* **Performance Overhead** (Chi phí hiệu năng): Mặc dù độ tin cậy của RabbitMQ là một điểm mạnh nhưng cơ chế xác nhận và lưu trữ có thể gây ra một số chi phí về hiệu năng, đặc biệt là trong các tình huống cần thông lượng cao.
* **Queues of Accumulation** (Hàng đợi tích luỹ): Nếu consumer không thể tiêu thụ kịp message dẫn tới tích lũy message trong queue. Điều này có thể gây ra tiêu tốn tài nguyên và có khả năng dẫn đến các vấn đề về hiệu suất.
* **Message Serialization** (Tuần tự hoá tin nhắn): Việc tuần tự hóa và giải tuần tự hóa các đối tượng phức tạp có thể tốn thời gian, ảnh hưởng đến hiệu suất tổng thể.

1. **Use case**

* **Some Use Cases:**
  + **Microservices Communication:** Passing events or commands between services without tight coupling.
  + **Task Offloading:** Background jobs such as email sending, report generation, or video processing.
  + **Event Streaming:** Real‑time analytics pipelines (e.g., using Kafka for high‑throughput event streaming).