**BÁO CÁO EX9\_MESSAGE QUEUE**

1. **Message queue**
2. Khái niệm

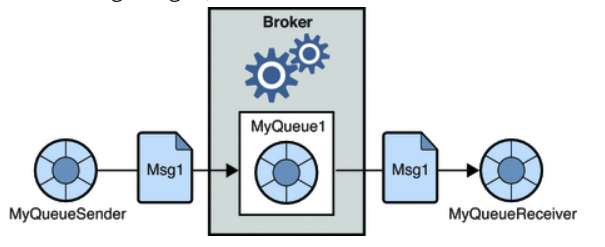
* Message queue (hàng đợi tin nhắn) là một hệ thống giao tiếp giữa các thành phần ứng dụng trong môi trường phân tán. Nó cho phép các ứng dụng gửi và nhận tin nhắn một cách bất đồng bộ, tức là không cần thiết lập trực tiếp kết nối giữa chúng.
* Trong môi trường phân tán, có nhiều thành phần hoạt động đồng thời và có thể sẽ cần truyền thông tin hoặc tương tác với nhau. Message queue giúp quán lý việc này một các hiệu quả bằng các lưu trữ các tin nhắn trong một hàng đợi tạm thời. Các thành phần ứng dụng có thể gửi tin nhắn vào hàng đợi và nhận tin nhắn từ hàng đợi một cách độc lập và không phụ thuộc và sự hiện diện của nhau.
* Ý nghĩa của queue ở đây chính là 1 hàng đợi chứa message chờ để được xử lý tuần tự theo cơ chế vào trước ra trước (FIFO – First In First Out).

1. Một số đặc điểm và lợi ích của Message queue bao gồm:

* Giảm sự phụ thuộc giữa các thành phần: Các thành phần ứng dụng có thể hoạt động độc lập và không cần biết về sự tồn tại của nhạu. Điều này giảm sự phụ thuộc và tăng tính linh hoạt của hệ thống.
* Hỗ trợ mô hình Publish-Subcribe: Message queue hỗ trợ mô hình Publish-Subcribe, cho phép một ứng dụng (publisher) gửi thông điệp và nhiều ứng dụng (subcribers) có thể nhận thôn điệp đó.
* Đảm bảo tính toàn vẹn và tin cậy: Message queue đảm bảo rằng các tin nhắn được gửi và nhận một cách tin cậy và không bị mất.
* Điều tiết tải: Message queue cho phép quản lý việc xử lý tin nhắn một cách linh hoạt và có thể được sử dụng để điều tiết tải cho các ứng dụng.
* Tích hợp các hệ thống không đồng nhất: Message queue giúp tích hợp các ứng dụng và hệ thống không đồng nhất, cho phép chúng giao tiếp với nhau thông qua một giao diện đơn giản là hàng đợi tin nhắn.

1. Các loại message queue

* Point-to-point: Message queue có thể là kiểu point-to-point, tức là khi đó ta chỉ có một hàng đợi và một cunsumer duy nhất để xử lý các tin nhắn trong hàng đợi.



* Publisher-Subcriber: Trong đó publisher (nhà sản xuất) gửi tin nhắn đến hàng đợi (trong trường hợp này được gọi là Topic) và tất cả các subcriber (người đăng ký) vào cùng 1 Topic đều sẽ nhận được tin nhăn trong Topic đó.

A diagram of a document

Description automatically generated

1. Message queue trong kiến trúc microservice

* Message queue là một thành phần quan trọng trong kiến trúc microservice. Trong kiến trúc này, ứng dụng được chia thành các dịch vụ nhỏ, độc lập và có thể triển khai, mở rộng riêng lẻ. Để kết nối và tương tác giữa các dịch vụ này, message queue được sử dụng như một cơ chế trung gian cho việc gửi và nhận các thông điệp.
* Các vai trò của message queue trong kiến trúc microservice:
* Giao tiếp giữa các dịch vụ: Message queue cho phép các dịch vụ trong kiến trúc microservice giao tiếp với nhau một các không đồng bộ và không phụ thuộc và sự hiện diện của dịch vụ khác. Thay vì gọi trực tiếp API của dịch vụ, một dịch vụ có thể gửi một tin nhăn tời message queue, và các dịch vụ khác có thể lắng nghe và xử lý tin nhắn đó một các độc lập.
* Phân tán và cân bằng tải: Message queue giúp phân tán cho các dịch vụ bằng các chia sẻ công việc và thông tin giữa các dịch vụ. Nó cho phép xử lý và phản hồi các yêu cầu có thể được phân tán một các hiệu quả giữa các dịch vụ.
* Xử lý các sự kiện và thông báo: Message queue hỗ trợ mô hình Publisher-Subcribe, cho phép các dịch vụ nhận thông báo và sự kiện từ các dịch vụ khác. Điều này giúp thực hiện các kịch bản như thông báo, gửi thông tin cập nhật hoặc thực hiện các hành động đáp ứng sự kiện.
* Bảo đảm tính vẹn toàn và tin cậy: Message queue đảm bảo rằng các thông điệp được gửi và nhận một các tin cậy và không bị mất. Nó hỗ trợ việc lưu trữ tạm thời các thông điệp để đảm bảo rằng các dịch vụ có thể xử lý thông điệp trong trường hợp xảy ra sự cố.
* Một số công nghệ message queue phổ biến được sử dụng trong kiến trúc microservice bao gồm Apache Kafka, RabbitMQ, Amazon SQS và Redis. Sử dụng message queue trong kiến trúc microservice giúp cải thiện tính linh hoạt, mở rộng và khả năng mở rộng của hệ thống và giảm sự phụ thuộc giữa các dịch vụ.

1. **RabbitMQ**
2. RabbitMQ là gì?

* RabbitMQ là một message queue mã nguồn mở được sử dụng phổ biến trong kiế trúc phân tán và hệ thống microservice. Nó được xây dụng trên giao thức AMQP (Advanced Message Queuing Protocol) và cung cấp một giải pháp linh hoạt và tin cậy để truyển thông tin giữa cá thành phần ứng dụng trong môi trường phân tán.
* RabbitMQ là một phần mềm trung gian được sử dụng để trao đổi dữ liệu các process, application, system hoặc server. RabbitMQ sẽ nhận message đến từ các thành phần khác nhau trong hệ thống, lưu trữ chúng an toàn trước khi đẩy đến đích.

1. Tại sao phải sử dụng RabbitMQ?
   1. Vấn đề

* Giả sử chúng ta có một web application cho phép user đăng ký thông tin. Sau khi đăng ký, hệ thống sẽ xử lý thông tin, generate PDF và gửi email lại cho user. Những công việc này hệ thông cần nhiều thời gian để xử lý, chẳng hạn. Nếu xử lý theo các thông thường khi user sẽ phải chờ đến khi hệ thông xử lý hoàn thành và nếu hệ thống có hàng nghìn user truy cập cùng lúc sẽ gây quá tải server.
* Đối với những loại công việc thế này chúng ta xử lý nó bằng các sử dụng message queue. Sau khi user nhập đầy đủ các thông trên web interface, web application sẽ tạo một Message “Generate PDF” chứa đầy đủ thông tin cần thiết và gửi nó vào Queue của RabbitMQ.



* Đây là một kiến trúc (architecture) cơ bản của message queue.
* Producer: là ứng dụng client, tạo message và publish tới broker.
* Consumer: là ứng dụng client khác, kết nối đến queue, subcriber và xử lý (consume) message.
* Broker (RabbitMQ): nhận message từ Producer, lưu trữ chúng an toàn trước khi được lấy từ Consumer.
* Mô hình giải quyết vấn đề trên:

A diagram of a cloud storage system

Description automatically generated

* (1) User gửi yêu cầu tạo PDF đến web application.
* (2) Web application (Producer) gửi tin nhắn đến RabbitMQ bao gồm dữ liệu từ request như tên và email.
* (3) Một trao đổi (Exchange) chấp nhận các tin nhắn từ Producer và định tuyến chúng đến Queue (hàng đợi) để tạo PDF.
* (4) Ứng dụng xử lý PDF (Consumer) nhận Message từ Queue và bắt đầu xử lý PDF.
* Một số vấn đề khác:
* Đối với các hệ thống sử dụng kiến trúc microservice thì việc gọi chéo giữa các service khá nhiều, khiến cho luồng xử lý khá phức tạp.
* Mức độ trao đổi data giữa các thành phần tăng lên khiến cho việc lập trình trở nên khó khăn hơn (vấn đề maintain khá đau đầu).
* Khi phát triển ứng dụng làm sao để các lập trình viên tập trung vào các domain, business logic thay vì các công việc trao đổi ở tầng infrastructure.
* Với các hệ thống phân tán, khi việc giao tiếp giữa các thành phần với nhau đòi hỏi chúng cần phải biết nhau. Nhưng điều này gây rắc rồi cho việc viết code. Một thành phần phải biết quá nhiều dẫn đến khó maintain, debug.
  1. Lợi ích và tính năng của RabbitMQ
* Transparency: Một producer không cần thiết phải biết consumer. Nó chỉ việc gửi message đến các queue trong message broker. Consumer chỉ việc đăng ký nhận message từ các queue này.
* Many Client: Vì producer giao tiếp với consumer trung gian qua message broker nên dù producer và consumer có khác biệt nhau về ngôn ngữ thì giao tiếp vẫn thành công. Hiện nay rabbitMQ đã hỗ trợ rất nhiều ngôn ngữ khác nhau.
* Asynchronous (bất đồng bộ): Producer không thể biết kho nào message đến được consumer hay khi nào message broker là xong việc. Consumer xử lý xong. Đối với producer, đẩy message đến massgae broker là xong việc. Consumer sẽ lấy message về khi nó muốn. Đặc tính này có thể được tận dụng để xây dụng các hệ thống lưu trữ và xử lý log.
* Flexible Routing: Message được định tuyến (route) thông qua Exchange trước khi đến Queue. RabbitMQ cung cấp một số loại Exchange thường dùng, chúng ta cũng có thể định nghĩa riêng Exchange cho riêng mình.
* Lightweight: một single instance của RabbitMQ chỉ chiểm khoảng 40MB RAM.
* Multiple message protocol: AMQP, MQTT, STOMP, HTTP.
* Cluster: các bạn có thể gom nhiều rabbitmq instance vào một cluster. Một queue được đinh nghĩa trên một instance khi đó đều có thể truy xuất từ các instance còn lại. Có thể tận dụng để làm load balancing.
* High availibilty: cho phép failover khi sử dụng mirror queue.
* Reliability: có cơ chế ACK để đảm bảo message được nhận bởi consumer đã được xử lý, lưu trữ (persistence) message, .high availability, publisher confirm, …
* Extensibility: cung cấp hệ thống plugin linh hoạt, dễ dàng tích hợp các plugin của third party. Ví dụ: plugin lưu message vào cơ sở dữ liệu.
* Cloud: dễ dàng triển khai với hạ tầng hiện có hoặc Cloud.
* Management & Monitoring: cung cấp HTTP API, command-line tool và UI để quản lý và giám sát.
* Tools support: Hoạt động với các công cụ CI/CD và có thể triển khai với BOSH, Chef, Docker và Puppet.

1. Những khái niệm cơ bản trong RabbitMQ

* Producer: Ứng dụng gửi message.
* Consumer: Ứng dụng nhận message.
* Queue: Lưu trữ message.
* Message: Thông tin truyền từ Producer đến Consumer qua RabbitMQ.
* Connection: Một kết nối TCP giữa ứng dụng và RabbitMQ broker.
* Channel: Một kết nối ảo trong một Connection. Việc publishing hoặc consuming message từ một queue đều được thực hiện trên channel.
* Exchange: Là nơi nhận message được publish từ Producer và đẩy chúng vào queue dữ vào quy tắc của từng loại Exchange. Để nhận được message, queue phải được nằm (binding) trong ít nhất 1 Exchange.
* Binding: Là quy tắc (rule) mà Exchange sử dụng để định tuyến Message đến Queue. Đảm nhận nhiệm vụ liên kết giữa Exchange và Queue.
* Routing key: Một key mà Exchange dữ vào đó để quyết định cách để định tuyến message đến queue. Có thể hiểu nôm na, Routing key là địa chỉ dành cho message.
* AMQP (Advence Message Queuing Protocol): là giao thức truyền message được sử dụng trong RabbitMQ.
* User: Để có thể truy cập vào RabbitMQ, chúng ta phải có username và password. Trong RabbitMQ, mỗi user được chỉ định với một quyền hạn nào đó. User có thể được phân quyền đặc biệt cho một Vhost nào đó.
* Virtual host/ Vhost: cung cấp những cách riêng biệt để các ứng dụng dùng chung một RabbitMQ instance. Những user khác nhau có thể có các quyền khác nhau đối với vhost khác nhau. Queue và Exchange có thể được tạo, vì vậy chúng chỉ tổn tại trong một vhost.

1. RabbitMQ hoạt động như thế nào?

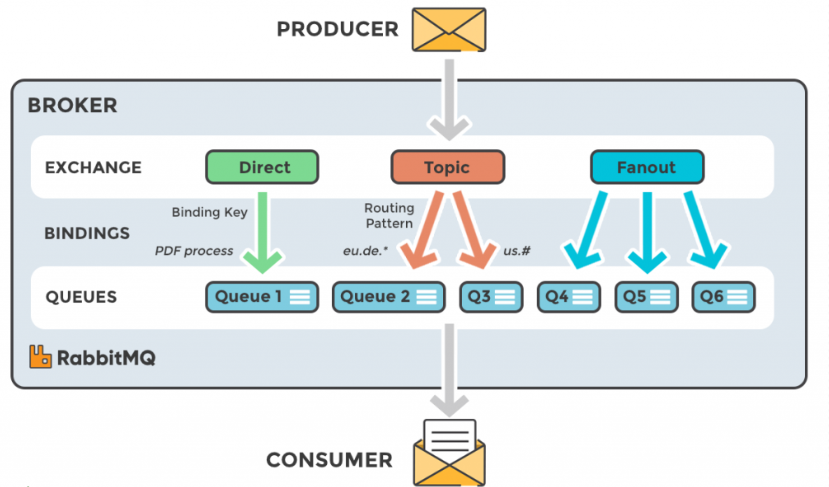
A diagram of a company

Description automatically generated

* (1) Producer đẩy message vào Exchange. Khi tạo Exchange, bạn phải mô tả nó thuộc loại gì. Các loại Exchange sẽ được giải thích phía dưới.
* (2) Sau khi Exchange nhận Message, nó chịu trách nhiệm định tuyến message. Exchange sẽ chịu trách nhiệm về các thuộc tính của message, ví dụ routing key, loại Exchange.
* (3) Việc binding phải được tạo ra từ Exchange đến Queue (hàng đợi). Trong trường hợp này, ta sẽ có hai binding đến hai hàng đợi khác nhau từ một Exchange. Exchange sẽ định tuyến Message vào các hàng đợi dữa trên thuộc tính của từng Message.
* (4) Các Message nằm ở hàng đợi đến khi chúng được xử lý bởi một Consumer.
* (5) Consumer xử lý Message nhận từ Queue.

1. Exchange và các loại Exchange

* Message không được publish trực tiếp vào Queue thay vào đó, Producer gửi message đến Exchange.
* Exchange là nơi mà các message được gởi bởi. Exchange nhận tin nhắn và định tuyến nó đến 0 hoặc nhiều Queue với sự trợ giúp của các ràng buộc (binding) và các khóa định tuyến (routing key).
* Thuật toán định tuyến được sử dụng phụ thuộc vào loại Exchange và quy tắc (còn gọi là ràng buộc hay binding).



* Có 4 loại Exchange: Direct, Fanout, Topic, Headers. Lựa chọn các Exchange type khác nhau sẽ dẫn đến các đối xử khác nhau của message broker với tin nhắn được từ producer. Exchange được bind (liên kết) đến một số Queue nhất định.

|  |  |
| --- | --- |
| **EXCHANGE TYPE** | **TÊN MẶC ĐỊNH** |
| Direct exchange | (Empty string) hoặc amq.direct |
| Fanout exchange | amq.fanout |
| Topic exchange | amq.topic |
| Headers exchange | amq.match (và amq.headers trong RabbitMQ) |

* Ngoài Exchange type, Exchange còn định nghĩa một số thuộc tính:
* Name: tên Exchange.
* Durability: thời gian tồn tại khi broker restart.
* Auto-delete: exchange bị xóa khi hàng đợi cuối cùng không còn binding từ nó.
* Arguments: các tham số không bắt buộc, được sử dụng bởi các plugin và các tính năng dành riêng cho broker.

1. **Kafka**
2. Kafka là gì?

* Kafka là một hệ thống mã nguồn mở được sử dụng để xây dựng các ứng dụng phân tán, chủ yếu dùng để xử lý, lưu trữ và chuyển tiếp luồn dữ liệu thời gian thực. Nó được phát triển bởi Apache Software Foundation và được đặt tên theo tên của nhà văn Franz Kafka.
* Apache Kafka hoạt động dữa trên mô hình thông điệp (messaging model) với các thành phần chính như nhà sản xuất (producer), dòng thông điệp (message stream), và người tiêu thụ (consume). Nó cho phép các ứng dụng và hệ thống gửi thông điệp và dữ liệu qua lại như một dòng chảy (stream), hỗ trợ việc phân tán, lưu trữ và xử lý dữ liệu từ nhiều nguồn khác nhau.
* Kafka thường được sử dụng trong các hệ thống xử lý dữ liệu lớn, nhưu hệ thống xử lý dòng dữ liệu thời gian thực (real-time streaming), hệ thông lưu trữ và truy vấn cơ sở dữ liệu và hệ thống phân tán với tính khả năng mở rộng cao. Nó cung cấp tính năng bền vững và đáng tin cậy, đảm bảo rằng dữ liệu được xử lý một cách hiệu quả và an toàn.
* Kafka đơn giản chỉ là một hệ thống gửi nhận message mục đích chính hay sử dụng là để giao tiếp giữa các hệ thống phân tán với nhau

A diagram of a mail connection

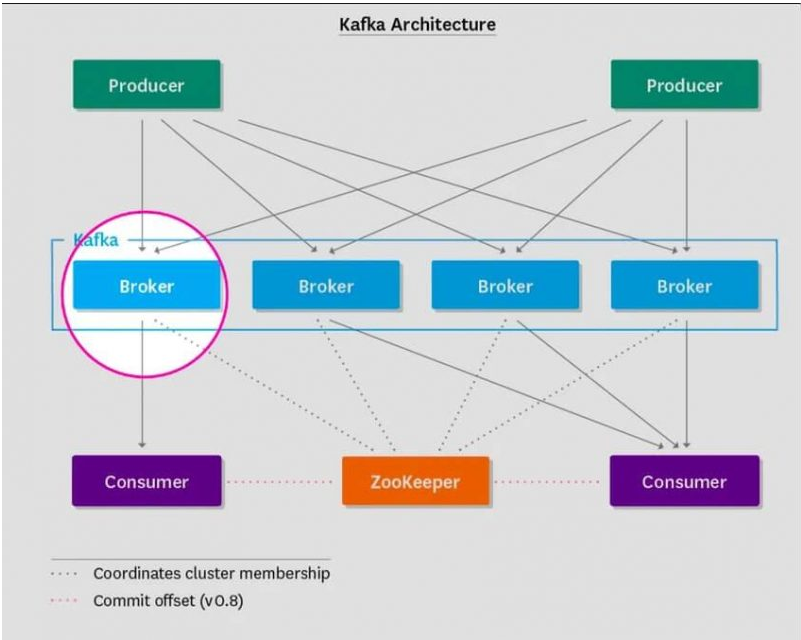
Description automatically generated

1. Một số thuật ngữ cơ bản trong Kafka

* Topic: Một dòng thông điệp trong Kafka, nơi các thông điệp được gửi và lưu trữ.
* Producer: Ứng dụng gửi các thông điệp (message) đến một hay nhiều topic trong Kafka.
* Consumer: Ứng dụng đọc và xử lý các thông điệp từ các topic trong Kafka.
* Broker: Một máy chủ hoạt động trong hệ thống Kafka, chịu trách nhiệm cho việc xử lý và lưu trữ thông điệp.
* ZooKeeper: Hệ thống quản lý cum (cluster management system) được sử dụng để cắt giảm và duy trì trạng thái của các nút (node) trong cụm Kafka.
* Kafka cung cấp một giải pháp mạnh mẽ và linh hoạt để xử lý dữ liệu trên quy mô lớn và thường được sử dụng trong các hệ thống dữa trên kiến trúc microservice và big data.

1. Apache Kafka hoạt động như thế nào?

* Kafka được xây dựng dựa trên mô hình publish/subcribe, tương tự như bất kỳ hệ thống message nào khác. Các ứng dụng (đóng vai trò là producer) gửi các message (records) tới một node kafka (broker) và nói rằng những message này được xử lý bởi các ứng dụng khác gọi là consumers.
* Các message được gửi tới kafka node sẽ được lưu trong một nơi gọi là topic và sau đó consumer có thể subcribe tới topic đó và lắng nghe những message này. Message có thể bất cữ thông tin gì như giá trị cảm biến, hành động người dùng.



1. **So sánh Kafka và RabbitMQ**
2. Kafka và RabbitMQ khác nhau ở điểm nào?

* Kafka và RabbitMQ là các hệ thống hàng đợi thông điệp được sử dụng trong quy trình xử lý luồng. Luồng dữ liệu gia tăng, liên tục và có khối lượng lớn, đòi hỏi phải xử lý ở tốc độ cao.

VD: Luồng dữ liệu có thể là dữ liệu cảm biến về môi trường cần liên tục thu thập và xử lý để có thể quan sát các thay đổ về nhiệt đọ hoặc khí áp theo thời gian thực.

* RabbitMQ là một trình tài nguyên tải thông điệp phân tán chịu trách nhiệm thu thập dữ liệu truyền phát từ nhiều nguồn nhằm khẳng định tuyến đến các đích đến khác nhau để tiến hành xử lý.
* Apache Kafka là một nền tảng truyền phát để xây dựng các ứng dụng truyền phát và quy trình dữ liệu thời gian thực. Kafka cung cấp một hệ thống tin nhắn có quy mô cực kỳ linh hoạt, khả năng chịu lỗi tốt và bền bỉ với nhiều tính năng hơn so với RabbitMQ.

1. Điểm khác biệt về kiến trúc giữa Kafka và RabbitMQ

* Cách tiếp cận kiến trúc RabbitMQ
* Một trình truyền tải RabbitMQ cho phép độ trễ thấp và phân bổ thông điệp phức tạp với các thành phần sau:
* Một cổng trao đổi nhận thông điệp từ đối tượng tạo và xác định xem chúng nên được định tuyến tới đâu.
* Một hàng đợi là kho lưu trữ nhận thông điệp từ một cổng trao đổi và gửi chúng đến đối tượng nhận.
* Một liên kết là một đường dẫn kết nối một cổng trao đổi và một trình truyền tải.
* Trong RabbitMQ, một khóa định tuyền là một thuộc tính thông điệp được sử dụng để định tuyến thông điệp từ một cổng trao đổi đến một hàng đợi cụ thể. Khi một đối tượng tạo gửi thông điệp đến công trao đổi, nó bao gồm một khóa định tuyến trong thông điệp. Cổng trao đổi sau đó sử dụng khóa định tuyến này để xác định hàng đợi mà thông điệp nên được gửi đến.
* Cách tiếp cận kiến trúc Kafka
* Một cum Kafka cung cấp khả năng xử lý sự kiện luồng có thông lượng cao với một kiến trúc phức tạp hơn. Đây là một thành phần chính của Kafka:
* Một trình truyền tải Kafka là một máy chủ Kafka cho phép các đối tượng tạo truyền dữ liệu đến các đối tượng nhận. Trình truyền tỉa Kafka chức các topic và phân vùng tương ứng của chúng.
* Một topic là một kho lưu trữ dữ liệu thực hiện nhóm các dữ liệu tương ứng trong một trình truyền tải Kafka.
* Một phân vùng là một kho lưu trữ dữ liệu nhỏ hơn trong một topic mà đối tượng nhận đăng ký nhận.
* ZooKeeper là phần mềm đặc biệt quản lý các cụm Kafka và phân vùng để cung cấp luồng có khả năng chịu lỗi. ZooKeeper gần đây đã được thay thế bằng giao thức Apache Kafka Raft (Kraft).
* Các đối tượng tạo trong Kafka gán một kháo thông điệp cho từng thông điệp. Sau đó trình truyền tải Kafka lưu trữ thông điệp trong phân vùng hàng đầu của topic cụ thể đó. Giao thức Kraft sử dụng các thuật toán đồng thuận để xác định phân vùng hàng đầu.

1. Kafka và RabbitMQ xử lý thông điệp khác nhau như thế nào?

* RabbitMQ và Apache Kafka di chuyển dữ liệu từ đối tượng tạo sang đối tượng nhận theo nhiều cách khác nhau. RabbitMQ là một trình truyền tải thông điệp đa dụng, ưu tiên phân phối thông điệp toàn diện. Kafka là một nền tảng truyền sự kiện phân tán, hỗ trợ việc trao đổi dữ liệu lớn liên tục theo thời gian thực.
* Xử lý thông điệp
* Trong Rabbit MQ, trình truyền tải đảm bảo rằng đối tượng nhận sẽ nhận được thông điệp. Đối tượng nhận có vai trò thụ động và chờ đợi trình truyền tải RabbitMQ đẩy thông điệp vào hàng đợi.
  + - VD: một ứng dụng ngân hàng có thể chờ thông điệp SMS cảnh bảo từ phần mềm xử lý giao dịch trung tâm.
* Tuy nhiên, đối tượng nhận Kafka chủ động hơn trong việc đọc và theo dõi thông tin. Khi thông điệp được thêm vào các tệp bản ghi vật lý, các đối tượng nhận Kafka theo dõi thông điệp cuối cùng mà chúng đã đọc và cập nhật bộ theo dõi thứ tự của mình sao cho phù hợp. Một bộ theo dõi thứ tự là một bộ đếm sẽ tăng lên sau khi đọc một thông điệp. Với Kafka, đối tượng tạo không biết được việc đối tượng nhận truy xuất thông điệp.
* Ưu tiên thông điệp
* Các trình truyền tải RabbitMQ cho phép phần mềm tạo thông điệp đẩy mức độ ưu tiên lên trước một số thông điệp nhất định bằng cách sử dụng hàng đợi ưu tiên. Thay vì gửi thông điệp theo thứ tự đến trước xử lý trước, trình truyền tải xử lý các thông điệp có độ ưu tiên cao hơn trước các thông điệp thông thường. Ví dụ: một ứng dụng bán lẻ có thể xếp hàng đợi các giao dịch bán hàng mỗi giờ. Tuy nhiên, nếu ứng dụng quản trị hệ thống phát một thông điệp cơ sở dữ liệu sao lưu ưu tiên, trình truyền tải sẽ gửi nó đi ngay lập tức.
* Khác với RabbitMQ, Apache Kafka không hỗ trợ hàng đợi ưu tiên. Kafka xử lý tất cả các thông điệp theo cách bình đẳng khi phân bổ chúng cho các phân vùng tương ứng của chúng.
* Thứ tự thông điệp
* RabbitMQ gửi và xếp hàng đợi các thông điệp theo một thứ tự cụ thể. Trừ khi một thông điệp ưu tiên cao hơn được xếp vào hàng đợi trong hệ thống, đối tượng nhận sẽ nhận được thông điệp theo thứ tự mà chúng được gửi.
* Trong khi đó, Kafka sử dụng các topic và phân vùng để xếp hàng đợi thông điệp. Khi một đối tượng tạo gửi đi một thông điệp, thông điệp này chuyển vào một topic và phân vùng cụ thể. Bởi vì Kafka không hỗ trợ trao đổi trực tiếp giữa đối tượng tạo-đối tượng nhận, đối tượng nhận sẽ lấy thông điệp từ phân vùng theo một thứ tự khác.
* Xóa thông điệp
* Một trình truyền tải RabbitMQ định tuyến thông điệp đến hàng đợi đích. Sau khi đọc, đối tượng nhận sẽ gửi trả lời xác nhận (ACK) cho trình truyền tải, rồi xóa thông điệp khỏi hàng đợi.
* Khác với RabbitMQ, Apache Kafka gắn thông điệp vào một tệp bản ghi, tệp này vẫn còn cho đến khi thời gian lưu giữ hết hạn. Bằng cách đó, đối tượng nhận có thể xử lý lại dữ liệu được truyền trực tiếp vào bất cứ lúc nào trong khoảng thời gian quy định.

1. Những điểm khác biệt chính giữa Kafka và RabbitMQ

* RabbitMQ cung cấp định tuyến thông điệp phức tạp với kiến trúc đơn giản, trong khi Kafka cung cấp một hệ thống truyền tải thông điệp bền bỉ cho phép các ứng dụng xử lý dữ liệu trong lịch xử lý luồng.
* Hiệu năng
* Cả RabbitMQ và Kafka đều cung cấp khả năng truyền tải thông điệp hiệu xuất cao cho các trường hợp sử dụng như dự kiến. Tuy nhiên, Kafka vượt trội hơn RabbitMQ về khả năng truyền thông điệp.
* Kafka có thể gửi hàng triệu thông điệp mỗi giây vì nó sử dụng I/O ổ đĩa tuần tự để cho phép trao đổi thông điệp có thông lượng cao. I/O ổ đĩa tuần tự là một hệ thống lưu trữ có chức năng lưu trữ và truy cập dữ liệu không gian bộ nhớ liền kề, và nó nhanh hơn so với truy cập ổ đĩa ngẫu nhiên.
* RabbitMQ cũng có thể gửi hàng triệu thông điệp mỗi giây, nhưng nó đòi hỏi phải có nhiều trình truyền tải đề làm được như vậy. Thông thường hiệu suất trung bình của RabbitMQ là khoảng hàng ngàn thông điệp mỗi giây và có thể chậm lại nếu hàng đợi của RabbitMQ bị tắc nghẽn.
* Bảo mật
* RabbitMQ và Kafka cho phép các ứng dụng trao đổi thông điệp một cách bảo mật nhưng bằng các công nghệ khác nhau.
* RabbitMQ đi kèm với các công cụ quản trị để quản lý quyền của người dùng và bảo mật trình truyền tải.
* Trong khi đó, kiến trúc Apache Kafka cung cấp các luồng sự kiện bảo mật nhờ LTS và java Authentication and Authorization Service (JAAS). TLS là một công nghệ mã hóa ngăn chặn việc nghe lén thông điệp không mong muốn và JAAS kiểm soát ứng dụng nào có quyền truy cập vào hệ thống truyền tải.

1. Những điểm tương đồng giữa Kafka và RabbitMQ là gì?

* Các ứng dụng cần trình truyền tải thông điệp đáng tin cậy để trao đổi dữ liệu trên đám mây. Cả RabbitMQ và Kafka đều cung cấp các nền tảng có quy mô linh hoạt và khả năng chịu lỗi để đáp ưng nhu cầu về lưu lượng truy cập ngày càng tăng và tính sẵn sàng cao.
* Khả năng điều chỉnh quy mô
* RabbitMQ có thể mở rộng khả năng xử lý thông điệp của mình cả theo chiều ngang lẫn chiều dọc. Bạn có thể phân bổ thêm tài nguyên điện toán cho máy chủ của RabbitMQ để tăng hiệu quả trao đổi thông điệp. Trong một số trường hợp, các nhà phát triển sử dụng một kỹ thuật phân phối thông điệp được gọi là trao đổi băm nhất quán RabbitMQ để xử lý cân bằng tải trên nhiều trình truyền tải.
* Tương tự như vậy, kiến trúc Kafka cho phép bổ sung thêm nhiều phân vùng vào một topic cụ thể để phân bổ tải thông điệp đồng đều.
* Khả năng chịu lỗi cao
* Cả Kafka và RabbitMQ đều là các kiến trúc xếp hàng đợi thông điệp mạnh mẽ, có khả năng phục hồi trước sự cố hệ thống.
* Bạn có thể nhóm nhiều trình truyền tải RabbitMQ thành các cụm và triển khai chúng trên các máy chủ khác nhau. RabbitMQ cũng sao chép các thông điệp xếp hàng đợi trên khắp các nút phân tán. Điều này cho phép hệ thống phục hồi từ sự cố gây ảnh hưởng đến bất kỳ máy chủ nào.
* Giống như RabbitMQ, Apache Kafka cũng có khả năng phục hồi và dự phòng tương tự bằng cách lưu trữ các cụm Kafka trên các máy chủ khác nhau. Mỗi cụm bao gồm các bản sao của tệp bản ghi mà bạn có thể phục hồi trong trường hợp xảy ra sự cố.
* Dễ sử dụng
* Cả hai hệ thống hàng đợi thông điệp đều có sự hỗ trợ mạnh mẽ từ cộng đồng và thư viện giúp việc gửi, đọc và xử lý thông điệp trở nên đơn giản. Điều này làm cho việc phát triển các ứng dụng khách trở nên dễ dàng hơn cho các nhà phát triển trên cả hai hệ thống.
* Ví dụ: bạn có thể sử dụng Kafka Streams (thư viện máy khách) để xây dựng hệ thống gửi thông điệp trên Kafka và Spring Cloud Data Flow để xây dựng các vi dịch vụ hướng sự kiện bằng RabbitMQ.

1. Tóm tắt các điểm khác biệt giữa Kafka và RabbitMQ

|  |  |  |
| --- | --- | --- |
|  | **RabbitMQ** | **Kafka** |
| Kiến trúc | Kiến trúc của RabbitMQ được thiết kế cho việc định tuyến thông điệp phức tạp. RabbitMQ sử dụng mô hình đẩy. Đối tượng tạo sẽ gửi các thông điệp đến đối tượng nhận kèm theo các quy tắc khác nhau. | Kafka sử dụng thiết kế dựa trên phân vùng để xử lý luồng thông lượng cao theo thời gian thực. Kafka sử dụng mô hình kéo. Các đối tượng tạo gửi thông điệp tới các topic và phân vùng mà đối tượng nhận đăng ký nhận. |
| Xử lý thông điệp | Các trình truyền tải RabbitMQ giám sát việc xử lý thông điệp. Các trình này sẽ xóa thông điệp sau khi thông điệp được xử lý. Trình truyền tải hỗ trợ các ưu tiên thông điệp. | Đối tượng nhận theo dõi việc truy xuất thông điệp bằng bộ theo dõi thứ tự. Kafka giữ lại các thông điệp theo chính sách lưu giữ. Không có ưu tiên thông điệp. |
| Hiệu năng | RabbitMQ có độ trễ thấp. RabbitMQ gửi hàng ngàn thông điệp mỗi giây | Kafka có khả năng truyền tải theo thời gian thực lên đến hàng triệu thông điệp mỗi giây. |
| Ngôn ngữ lập trình và giao thức | RabbitMQ hỗ trợ một loạt các ngôn ngữ và giao thức cũ. | Kafka có số lựa chọn hạn chế về ngôn ngữ lập trình. Kafka sử dụng giao thưc nhị phân qua TCP để truyền dữ liệu. |