Tìm hiểu về Microservices và các công nghệ liên quan

I Microservice là gì?

- Mỗi dịch vụ sẽ được chia nhỏ thành nhiều thành phần khác nhau, mỗi thành phần sẽ hoạt động độc lập, được phát triển độc lập và chỉ xử lý các nghiệp vụ chức năng của nó. Mỗi thành phần cũng sẽ không lệ thuộc vào công nghệ phát triển với các thành phần khác.

1 Ưu điểm

- Các service có thể được bảo trì độc lập.
- Dễ dàng nắm bắt các business logic.
- Có thể áp dụng được nhiều công nghệ mới.
- Dễ dàng scale khi hệ thống trở nên phức tạp.
- Bảo mật tối ưu cho source code.

2 Nhược điểm

- Có độ trễ cao.
- Khó phát triển.
- Khó đảm bảo tính toán vẹn dữ liệu.

3 Tại sao nên dùng?

So với kiến trúc **Microservice** thì kiểu kiến trúc **Monolithic** thông thường sẽ gây ra một số bất lợi:

- Khó để có thể áp dụng triển khai theo mô hình agile.
- Nếu service không ổn định có thể sẽ làm sập cả hệ thống.
- Sẽ triển khai toàn bộ hệ thống mặc dù chỉ phải cập nhật hoặc nâng cấp một phần nhỏ.
- Được thiết kế, phát triển và triển khai dựa theo một khối duy nhất nên sẽ có nhược điểm nhất định.
- Gây phức tạp và gặp nhiều khó khăn trong quá trình nâng cấp, bảo trì hoặc thêm các tính năng mới.
- Các ứng dụng Monolithic cần phải sử dụng chung một công nghệ nên khó để thay đổi hay áp dụng thêm các công nghê mới.

II Công nghệ sử dụng trong Microservice

1 Docker

1.1 Khái niệm

- Sử dụng docker giúp đơn giản hóa việc setup môi trường, đóng gói các service và deploy hệ thống.

2 RabbitMQ:

2.1 Khái niệm

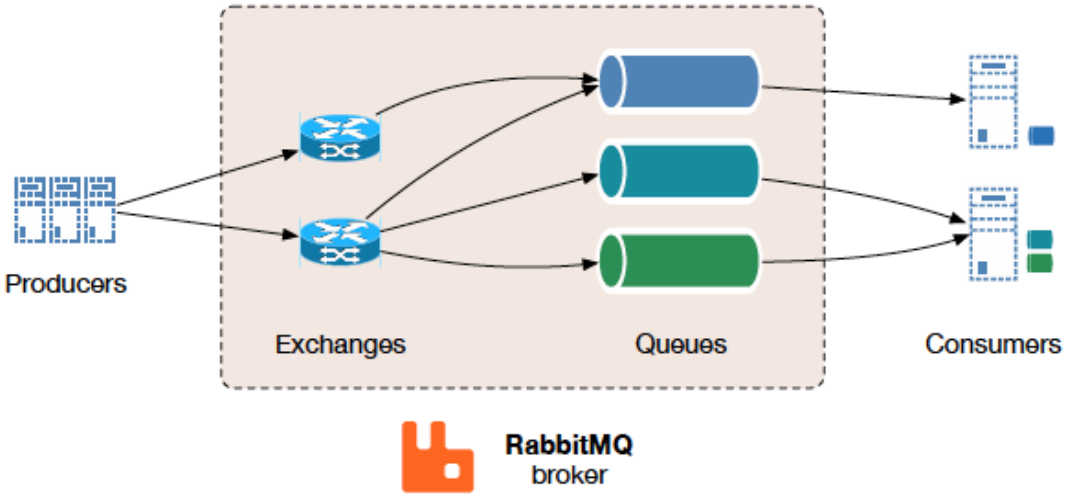
- Là message broker mã nguồn mở, có dung lượng nhẹ, dễ dàng triển khai trên nhiều hệ điều hành được phát triển để giao tiếp, trao đổi dữ liệu giữa các ứng dụng, hệ thống hay server khác nhau một cách dễ dàng.

2.2 Nhiệm vụ

Message broker sẽ đảm nhiệm việc validating (xác thực), transforming (chuyển đổi) và routing (định tuyến) cho messages giữa các ứng dụng với nhau.

2.3 Hoạt động

Producer: gửi một message ➔ **server process**: nhận message thực hiện các nhiệm vụ (validating, transforming, routing, persistence, delivery, …) ➔ **Consumer**: nhận message.



2.4 Lợi ích:

Asynchronous Messageing

- Hỗ trợ rất nhiều giao thức message như queue, delivery, routing, …

Developer Experience:

- Tính linh hoạt cao chỉ k tương thích với ngôn ngữ Erlang, còn lại có khả năng tương thích hầu hết các ngôn ngữ lập trình thông dụng.

Distibuted Deployment

- Tính khả dụng cao và thông lượng lớn, RabbitMQ có thể triển khai dưới dạng cluster.

Enterprise & Cloud Ready

- Với các công nghệ authentication, authorization được phát triển dựa trên TLS và LDAP gia tăng độ bảo mật lên rất cao.

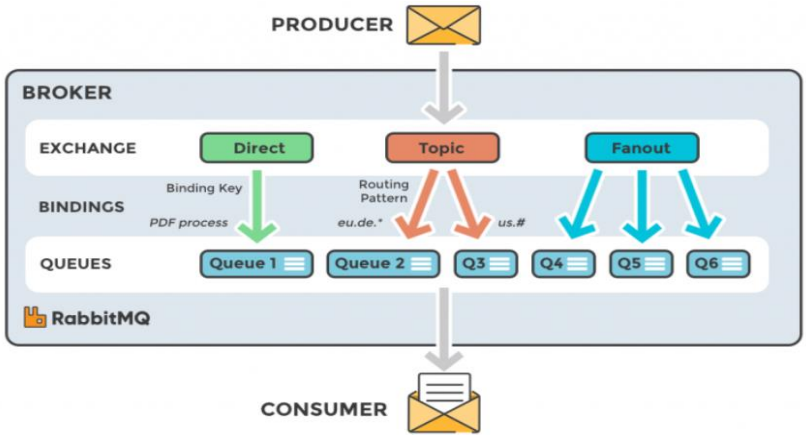
Management & Monitoring

- Có thể sử dụng HTTP-API để quản lý và giám sát RabbitMQ một cách hiệu quả nhất.

Tool & Plugins

- Được cung cấp nhiều tool và plugin được phát triển liên tục với khả năng như tích hợp, đo lương và tương thích với các hệ thống khác của doanh nghiệp.

2.5 Các loại Exchange trong RabbitMQ

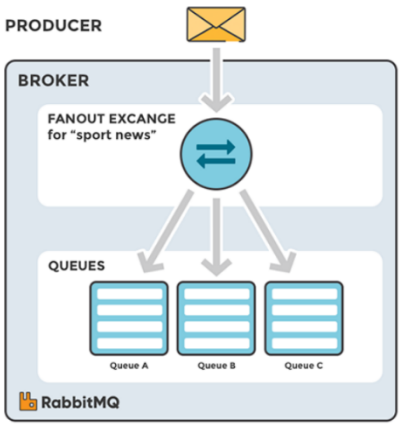


Direct exchange

- Là loại phổ biến nhất, message sẽ được truyền trực tiếp đến queue dựa trên routing key.

VD: Nếu hàng đợi exchange có blinding key là create_PDF thì message có routing key là create_PDF sẽ được đẩy vào hàng đợi của exchange đó.

Fanout exchange



- Message sẽ được đẩy đến toàn bộ queue mà nhận nó.

Default exchange

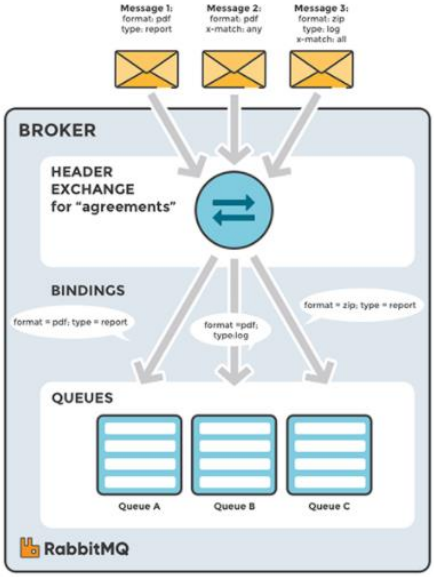
- Message sẽ được truyền thẳng tới queue có tên trùng với routing key trên message

Topic exchange

- Message được vận chuyển tới queue dựa vào routing key nhưng mức độ khác rộng rãi nó sẽ đánh giá xem 2 routing key có match với nhau hay không rồi mới chuyển tới queue.

VD: ‘’rabbit” và “rabit.dev” là 2 routing key match với nhau vậy nên tin nhắn sẽ được chuyển đến queue.

Header exchange

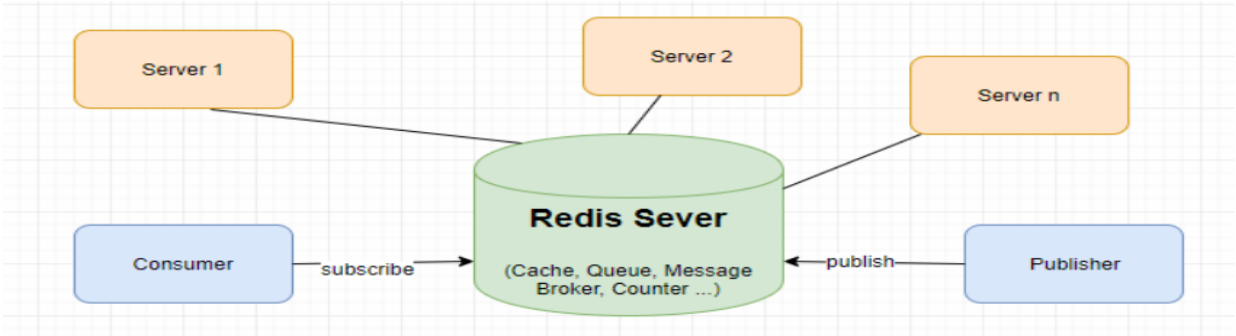


- Mỗi queue khi khai báo message từ exchange sẽ phải khai báo rõ là nhận message có phần header như thế nào, bao gồm những trường và từ khóa. Exchange sẽ dựa vào tin message và các điều kiện đó mà phân phối tin nhắn cho phù hợp.

3 Redis

3.1 Khái niệm

- Là một mã nguồn mở dùng để lưu trữ dữ liệu có cấu trúc, có thể sử dụng như một DB, bộ nhớ cache hay một message broker.



3.2 Ứng dụng

Caching

- Sử dụng làm bộ nhớ đệm, nhờ tốc độ đọc ghi nhanh nên Redis có thể làm bộ nhớ đệm, nơi chia sẽ dữ liệu giữa các ứng dụng, hoặc làm DB tạm thời.

Counter:

- Sử dụng làm bộ đếm. với thuộc tính tăng giảm thông số rất nhanh trong khi dữ liệu được lưu trên RAM, sets và sorted sets được sử đụng để đếm lượt view của một website, các bảng xếp hạng game. Redis có hỗ trợ thread safe do đó nó có thể đồng bộ dữ liệu giữa các request.

Publish/Suscribe:

- Tạo kênh chia sẻ dữ liệu. Redis hỗ trwoj các channel để trao đổi dữ liệu giữa các publist/suscribe.

Queues:

- Tạo hàng đợi để xử lý lần lượt các request. Redis cho phép lưu trữ theo list và cung cấp rất nhiều thao tác với các phần từ trong list nên vì vậy nó có thể sử dụng như mộ message queue.

3.3 Persistent redis

Ngoài lưu trữ Key-value trên RAM, thì redis có 2 background threads chuyên làm nhiệm vụ định kỳ ghi dữ liệu trên đĩa cứng

3.3.1 RDB (Redis Database file)

- Thực hiện tạo và sao lưu snapshot của DB vào ổ cứng sau mỗi khoảng thời gian nhất đinh.

Ưu điểm:

- Cho phép lưu các version khác nhau của DB, rất thuận tiện khi xảy ra sự cố.
- Với việc lưu trữ data vào 1 file cố định, người dùng có thể dễ dàng chuyển data đến các data centers, máy chủ khác nhau.
- Khi restart server, dùng RDB làm việc với lượng lớn data lớn sẽ có tốc độ cao hơn là dùng AOF.

Nhược điểm:

- Khả năng mất mát dữ liệu cao hơn. Thông thường sẽ setup taoj RDB 5 phút 1 lần (hoặc có thể nhiều hơn), nên khi có sự cố, Redis không thể hoạt động, dữ liệu trong những phút cuối sẽ bị mất.

3.3.2 AOF (Append Only File)

- Lưu lại tất cả các thao tác write mà server nhận được, các thao tác này sẽ được chạy lại khi restart hoặc tái thiết lập dataset ban đầu.

Ưu điểm:

- Đảm bảo dataset được bền vững hơn so với dùng RDB, người dùng có thể config để Redis khi log theo từng câu query hoặc mỗi giây 1 lần.

N***hược điểm:***

- File AOF thường lớn hơn file RDB với cùng 1 dataset.
- Tốc độ của AOF có thể chậm hơn RDB tùy theo cách thức thiết lập khoảng thời gian cho việc sao lưu vào ổ cứng.
- Có thể gặp một số lỗi không thể tái tạo lại chính xác dataset khi restart redis khi sử dụng AOF.

4 GRPC

4.1 Khái niệm:

- Là một RPC framework giúp kết nối giữa các service trong hệ thống, hỗ trợ load balancing, tracing, health checking và authentication, hỗ trợ từ ứng dụng mobile, trình duyệt cho tới back-end service và có thể chạy trên bất kỳ môi trường nào và thường được sử dụng trong mô hình Microservice.

4.2 Tại sao nên dùng?

- Trong mô hình **Monolithic** thì việc sử dụng REST API giao tiếp giữa client và server đã được giải quyết khá tốt. Nhưng trong mô hình Microservice để có thể tăng tải và thông lượng giữa các services với tốc độ cao nhất, giảm tải quá trình encode/decode nên GRPC ra đời để giúp giải quyết những vấn đề đó.

4.3 Hoạt động

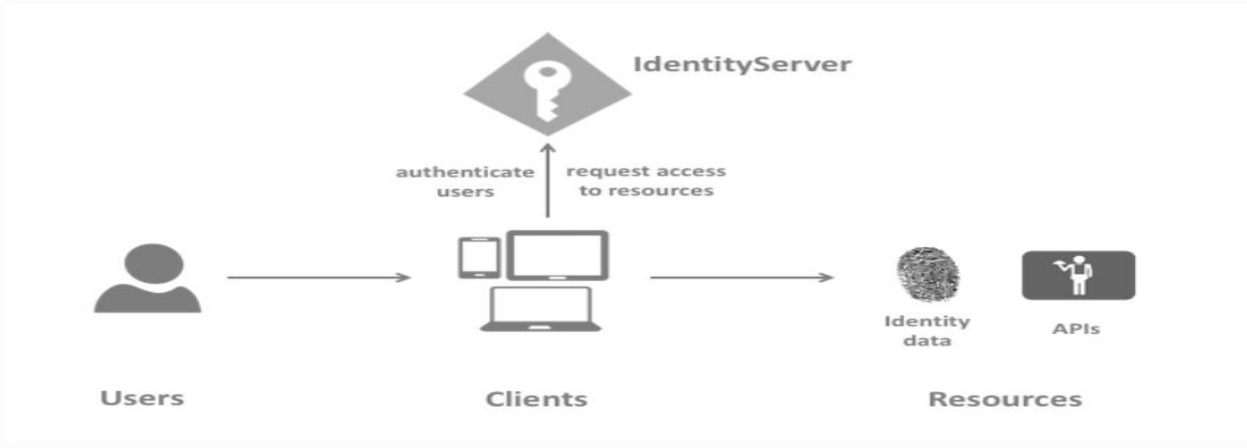
4.4 Lưu ý

- GRPC nên dùng cho giao tiếp giữa backend to backend. CPU không phải gánh nhiều cost cho encode/decode mỗi đầu.

5 IdentityServer4

5.1 Khái niệm

- là một framework hỗ trợ OpenID Connect và OAuth 2.0 trên ASP.NET Core. Nó cũng là một gói thư viện trên Nuget được dùng trong ASP.NET Core như một middleware cho phép đăng nhập/đăng xuất, cấp token, chứng thực và các giáo thức chuẩn khác.



- **User**: Phía người dùng.
- **Client**: Phía ứng dụng.
- **Resources**: Các API cần bảo vệ bởi IdentityServer4
- **Access Token**: Token được sử dụng để truy cập vào API Resource
- **Refresh Token**: mỗi một token đều có một thời gian hết hạn. Refresh token là việc lấy lại token mới mà không cần tương tác của người dùng. Client nên cho phép làm điều này bằng cách setup AllowOfflineAccess giá trị là true ở phần cài đặt client trong IdentityServer4.
- **Grant Type**: nó là loại tương tác giữa client và IdentityServer. Dựa trên client, có thể chọn loại grant type phù hợp.

6 MongoDB

6.1 Khái niệm:

Là một dạng NoSQL database, sử dụng lưu trữ dữ liệu dưới dạng Document JSON nên mỗi collection sẽ có kích thước và các document khác nhau. Dữ liệu được lưu dưới dạng JSON nên sẽ được truy vấn rất nhanh.

Ưu điểm:

- Dữ liệu lưu trữ phi cấu trúc, không có tính ràng buộc, toàn vẹn nên tính sẵn sàng cao, hiệu suất lớn và dễ dàng mở rộng lưu trữ.
- Dữ liệu được caching (ghi đệm) lên RAM, hạn chế truy cập vào ổ cứng nên tốc độ đọc và ghi cao.

Nhược điểm:

- Không ứng dụng được cho các mô hình giao dịch nào có yêu cầu độ chính xác cao do không có ràng buộc.
- Không có cơ chế transaction (giao dịch) để phục vụ các ứng dụng ngân hàng.
- Dữ liệu lấy RAM làm trọng tâm hoạt động vì vậy khi hoạt động yêu cầu một bộ nhớ RAM lớn.
- Mọi thay đổi về dữ liệu mặc định đều chưa được ghi xuống ổ cứng ngay lập tức vì vậy khả năng bị mất dữ liệu từ nguyên nhân mất điện đột xuất là rất cao.

7 SQL Server

7.1 Khái niệm:

- Là một hệ thống quản lý cơ sở dữ liệu quan hệ sử dụng SQL là ngôn ngữ cơ sở dữ liệu, được dùng để tạo, lấy xóa và sửa đổi

Ưu điểm:

- Cài nhiều phiên bản MS SQL khác nhau trên cùng một máy.
- Duy trì riêng biệt các môi trường sản xuất, phát triển, thử nghiệm.
- Giảm thiểu các vấn đề tạm thời trên cơ sở dữ liệu.
- Tách biệt các đặc quyền bảo mật.
- Duy trì máy chủ dự phòng.

Nhược điểm:

- Chỉ chạy trên hệ điều hành Windows.
- Cần thanh toán phí license để chạy nhiều database.