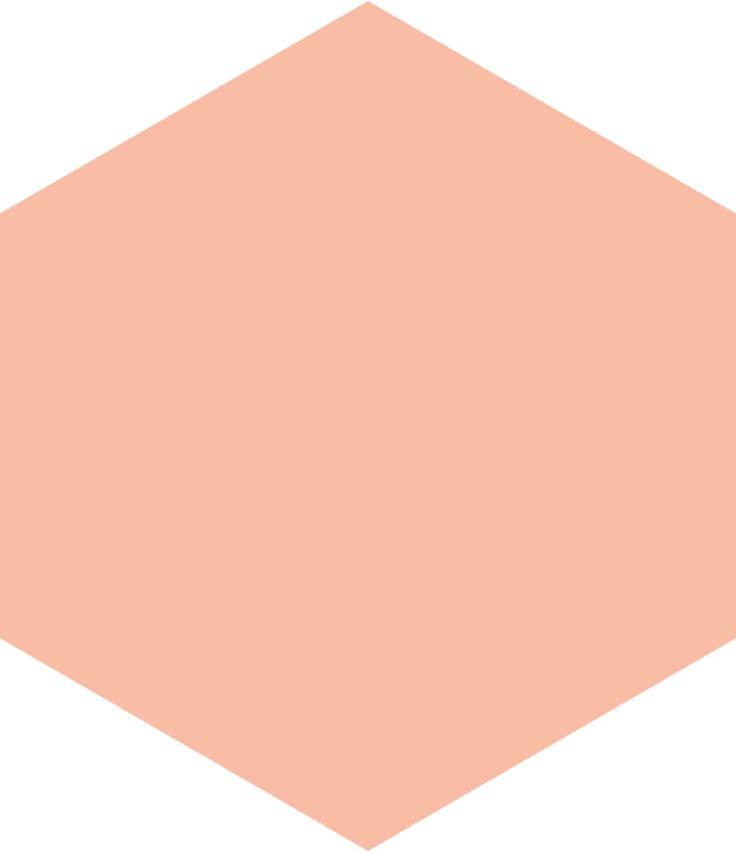
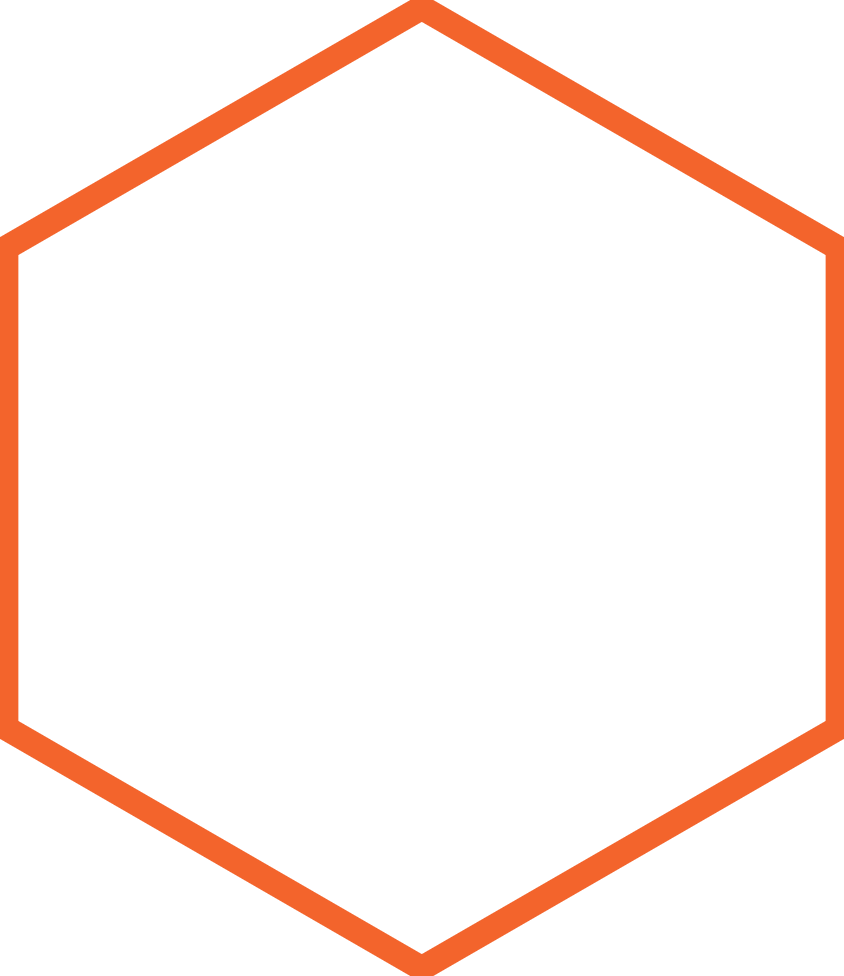


|  |
| --- |
| **Báo cáo Tìm hiểu gRPC** |
| Kiến trúc Hướng dịch vụ |
| Thành viên:  Phạm Văn Trọng Vũ Ngọc Chi Nguyễn Văn Minh |
|  |





Mục lục

[**I.** **Tổng quan về gRPC** 2](#_Toc3997729)

[1. gRPC là gì ? 2](#_Toc3997730)

[2. Cải tiến của gRPC so với các công nghệ trước đó 2](#_Toc3997731)

[3. Các kịch bản (trường hợp) sử dụng chính: 4](#_Toc3997732)

[4. Các bước sử dụng chính 5](#_Toc3997733)

[5. Phần kết luận 5](#_Toc3997734)

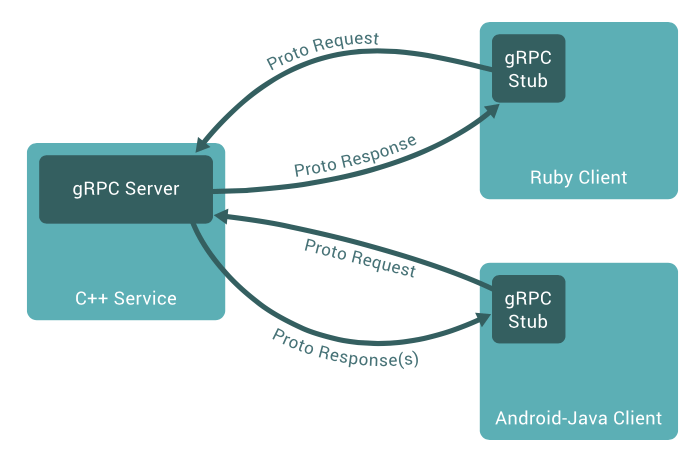
# **Tổng quan về gRPC**

## gRPC là gì ?

*gRPC* - gRPC Remote Procedure Call: là một framework RPC hiệu suất cao mã nguồn mở hiện đại có thể chạy trong mọi môi trường. Nó có thể kết nối hiệu quả các dịch vụ trong và trên các trung tâm dữ liệu lớn tạo ra sự hỗ trợ để cân bằng tải, theo dõi, kiểm tra và xác thực. Nó cũng được áp dụng trong bước cuối của các hệ thống điện toán phân tán để kết nối các thiết bị, ứng dụng di động và trình duyệt với các dịch vụ phụ trợ.

## Cải tiến của gRPC so với các công nghệ trước đó

*gRPC* là sự tiếp nối của khái niệm RPC, cho phép một tiến trình gọi liền mạch các phương thức của một máy từ xa, tương tự như lời gọi thủ tục thông thường cục bộ và ẩn đi việc truyền dữ liệu qua mạng



*gRPC* lấy và cải thiện ý tưởng từ nhiều công nghệ trước đó:

* Giống như CORBA, *gRPC* sử dụng Interface Definition Language (IDL) - Ngôn ngữ định nghĩa giao diện để xác định giao kèo dịch vụ.
* Giống như Java RMI, các máy khách *gRPC* gọi các phương thức từ xa như thể chúng đang chạy trên chính máy đó.
* Giống như SOAP, các công cụ *gRPC* có thể tự động tạo mã nguồn cho máy khách và máy chủ trong phạm vi một số ngôn ngữ.
* Giống như REST, *gRPC* sử dụng giao thức HTTP (cụ thể là HTTP / 2) làm cơ chế truyền tải.

Tuy nhiên, *gRPC* không chỉ là một tập hợp các công nghệ liên quan lỏng lẻo, mà là một framework mạnh, được thiết kế có chủ ý để tạo ra các kiến ​​trúc dịch vụ bền vững, từ việc scale quy mô các trung tâm dữ liệu đến các thiết bị IoT có kích thước nhỏ. *gRPC*:

* Hỗ trợ truyền phát hai chiều giữa máy khách (client) và máy chủ (server).
* Sử dụng Bộ đệm giao thức (*Protocol Buffers*) và định dạng dưới dạng mã nhị phân, tạo ra sự hiệu quả trong việc giữ cho request & respone nhẹ và nhanh (thay vì sử dụng XML hoặc JSON – hai định dạng dựa trên văn bản).
* Sử dụng giao thức HTTP / 2 (được phát triển bởi chính Google, nâng cấp nhiều so với phiên bản HTTP/1.1 hiện tại) để ghép các kết nối TCP tồn tại lâu dài thành nhiều kênh, góp phần tăng hiệu quả liên lạc và đảm bảo tính chất đồng thời giữa các bên.
* Có API trung gian mở rộng (*extensive middleware API*) để thực hiện các công việc liên quan đến bảo mật hoặc về cấu trúc như xác thực (authentication) , ủy quyền (authorization), kiểm soát luồng (flow-control), theo dõi (tracing), ngắt mạch (circuit-breaking) và chính sách dịch vụ (service policies).

*gRPC* cũng là một framework đa nền tảng. Theo Google công bố, *gRPC* hỗ trợ lên đến 10 ngôn ngữ, và được maintain với khả năng tương đương giữa các ngôn ngữ.

Điều này có nghĩa là một dịch vụ có thể được viết bằng Python tại một máy và *gRPC* có thể tạo ra các stub ở các máy khác để truy cập dịch vụ đó bằng bất kỳ ngôn ngữ nào trong các ngôn ngữ được hỗ trợ.



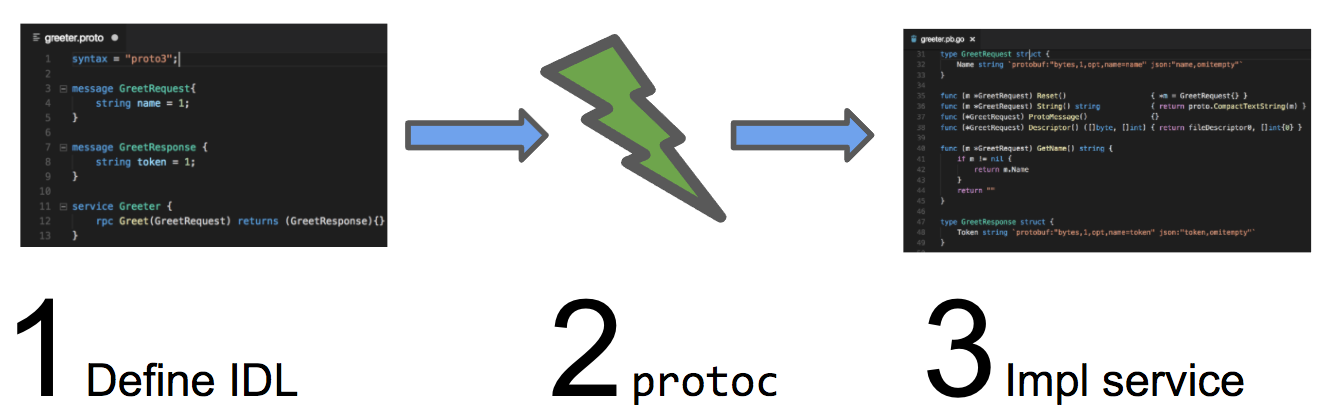
Các ngôn ngữ mà *gRPC* hỗ trợ (theo *gRPC.io*)

## Các kịch bản (trường hợp) sử dụng chính:

* Kết nối một cách hiệu quả các dịch vụ polyglot trong kiến trúc kiểu microservice.
* Kết nối các thiết bị di động, ứng dụng trình duyệt trên client với các dịch vụ phía sau (backend)
* Tạo các thư viện client hiệu quả

## Các bước sử dụng chính

Trong hầu hết các ngôn ngữ, việc tạo dịch vụ gRPC có thể được chắt lọc theo ba bước cơ bản:



Bước 1: Trước tiên, bạn xác định file bộ đệm giao thức (protocol buffers) IDL có chứa các hợp đồng dịch vụ.

Bước 2: Tiếp theo, sử dụng trình biên dịch bộ đệm giao thức ***protoc*** *(do rGPC cung cấp)* để tạo giao diện máy chủ và stub trên các máy khách với ngôn ngữ bạn chọn.

Bước 3: Cuối cùng, triển khai (implement) các phương thức cho các giao diện dịch vụ như cách chúng được định nghĩa trong file IDL của bạn.

II. Demo

Server : Java

Client 1: Python

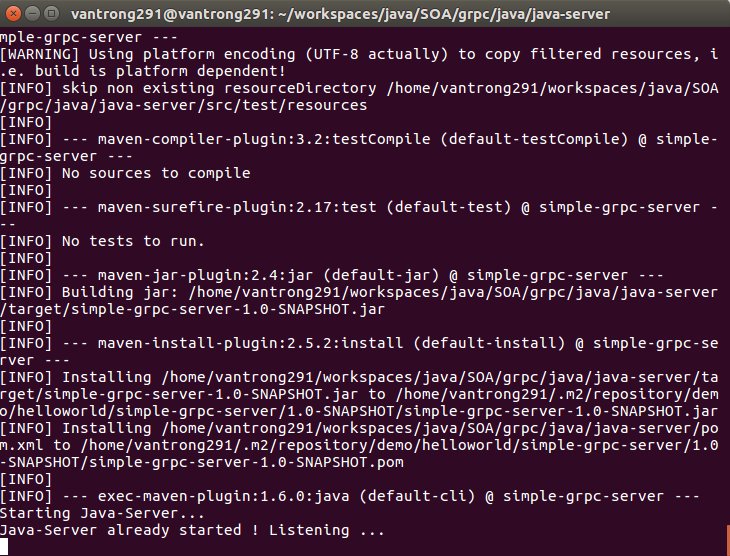
Client 2: Ruby

Bước 1: Run server:

– Cài đặt maven

– Di chuyển tới thư mục “java”. Chạy “mvn install”

- Di chuyển tới java-server. Chạy: “mvn install exec:java -Dexec.mainClass=grpc.greeter.server.JavaGrpcServer”



Bước 2: Run Client Python

– Cài pip (>9)

– sudo pip install grpcio grpcio-tools

- Di chuyển đển python3-client/helloworld

Run: ”python3 greeter\_client.py

