

Flutter Communication gRPC



Peter Kassenaar –
info@kassenaar.com



Communicating via gRPC

A possible way of communication inside your app with a backend

Communication – lots of options!



API Architecture Styles

ByteByteGo.com

Style	Illustration	Use Cases
SOAP		XML-based for enterprise applications
RESTful		Resource-based for web servers
GraphQL		Query language reduce network load
gRPC		High performance for microservices
WebSocket		Bi-directional for low-latency data exchange
Webhook		Asynchronous for event-driven application

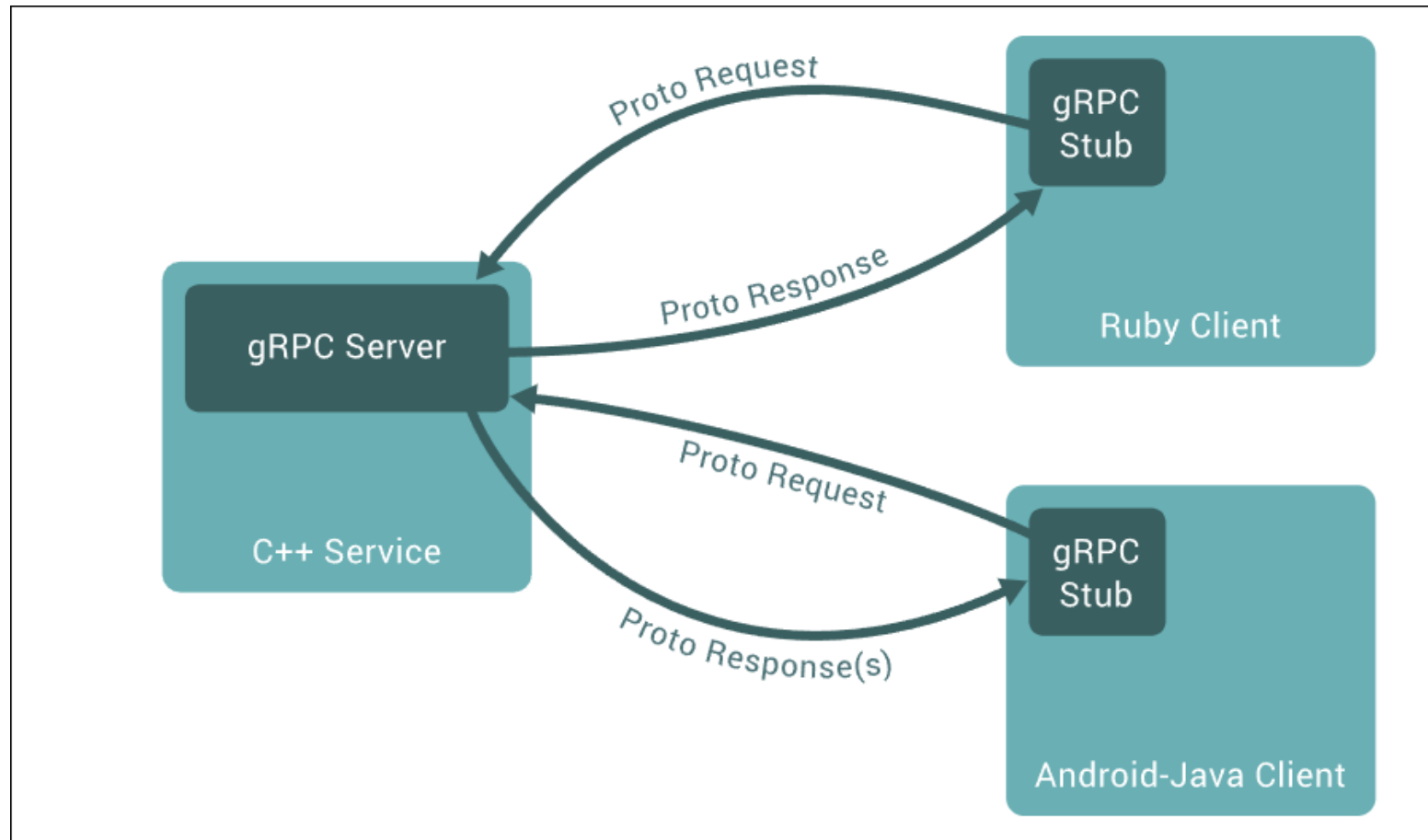
What is gRPC?



- gRPC is 'just' one of the possible **forms of communication**
 - It is an **implementation detail**.
 - gRPC CAN (in theory) be swapped out for **other types of communication** below the bloc architecture
- *"A high performance, open source universal RPC framework"*
- <https://grpc.io/>
- Basically **a wrapper** around `http`, using `http/2` protocol
- **Binary format** (as opposed to `json` and `xml`)



Introduction to gRPC



<https://grpc.io/docs/what-is-grpc/introduction/>

“gRPC in 5 minutes”



The video player shows the title "gRPC in 5 minutes" and the Google Cloud Platform logo. Below the title, two speakers are listed: Ivy Zhuang (Software Engineer, gRPC) and Eric Anderson (Software Engineer, gRPC). The video progress bar indicates 0:04 / 5:00. The video player interface includes standard controls like play, pause, volume, and full screen.

Google Cloud Platform | gRPC

gRPC in 5 minutes

Ivy Zhuang (she/her)
Software Engineer, gRPC
@yifeizhuang

Eric Anderson (he/him)
Software Engineer, gRPC
@ejona86

0:04 / 5:00

gRPC in 5 minutes | Eric Anderson & Ivy Zhuang, Google

<https://www.youtube.com/watch?v=njC24ts24Pg>

gRPC Prerequisites



- gRPC has multiple elements:
- **SERVER** – server responds to gRPC calls and sends data
- **CLIENT** – your app. Uses gRPC services to communicate with the backend
- **Dart** – as the programming language
- **Protocol buffer compiler** `protoc` ('protobuf')
- Dart **plug-in** for the compiler `protoc-gen-dart`

Protoc - compiler



- Needed to compile `.proto` files in your app
- Steps: protobuf.dev/installation/

Protocol Buffers Documentation

- Protocol Buffers
- Overview
- Protoc Installation**
- News
- Programming Guides
 - Language Guide (editions)
 - Language Guide (proto 2)
 - Language Guide (proto 3)
 - Proto Limits
 - Style Guide
 - Enum Behavior
 - Encoding
 - ProtoJSON Format
 - Techniques
 - Add-ons
 - Extension

Protoc Installation

Protocol Buffer Compiler Installation

How to install the protocol buffer compiler.

The protocol buffer compiler, `protoc`, is used to compile `.proto` files, which contain service and message definitions. Choose one of the methods given below to install `protoc`.

Install Pre-compiled Binaries (Any OS)

To install the latest release of the protocol compiler from pre-compiled binaries, follow these instructions:

1. From <https://github.com/google/protobuf/releases>, manually download the zip file corresponding to your operating system and computer architecture (`protoc-<version>-<os>-<arch>.zip`), or fetch the file using commands such as the following:

```
PB_REL="https://github.com/protocolbuffers/protobuf/releases"
curl -LO $PB_REL/download/v< param protoc-version >/protoc-< param protoc-version >-linux-
```




Example – the Hello World demo

Getting started with gRPC Dart server & client

Quick Start online (warning – going fast!)



- <https://grpc.io/docs/languages/dart/>

The screenshot shows the gRPC website's navigation and content for the Dart language. The left sidebar contains a search bar and a list of documentation categories. The 'Dart' category is highlighted with a red box, and its sub-items, 'Quick start' and 'Basics tutorial', are also visible. The main content area displays the 'Dart' title and two primary links: 'Quick start' (with the subtext 'Run your first Dart gRPC app in minutes!') and 'Basics tutorial' (with the subtext 'Learn about Dart gRPC basics.'). Below these, there are sections for 'Learn more' (Examples), 'Reference' (API), and 'Other' (grpc-dart repo, pub package). The footer indicates the page was last modified on December 18, 2020, and provides a link to add missing headings to the Table of Contents.

gRPC

About Meet Docs Guides Videos Showcase Blog Community

Search this site..

Docs

What is gRPC?

Languages

C# / .NET

C++

Dart

Quick start

Basics tutorial

API

Go

Java

Kotlin

Node

Objective-C

Docs / Languages / Dart

Dart

[Quick start](#)
Run your first Dart gRPC app in minutes!

[Basics tutorial](#)
Learn about Dart gRPC basics.

Learn more
[Examples](#)

Reference
[API](#)

Other
[grpc-dart repo](#)
[pub package](#)

Last modified December 18, 2020: [Add new shortcode to add missing headings to ToC \(#577\) \(8e4f008\)](#)

Installing Protobuf on windows



- `winget install protobuf`
- Other OS's: see protobuf.dev/installation/

```
Windows PowerShell
PS C:\Users\info> winget install protobuf
The 'msstore' source requires that you view the following agreements before using.
Terms of Transaction: https://aka.ms/microsoft-store-terms-of-transaction
The source requires the current machine's 2-letter geographic region to be sent to the
properly (ex. "US").

Do you agree to all the source agreements terms?
[Y] Yes [N] No: y
Found protobuf [Google.Protobuf] Version 29.3
This application is licensed to you by its owner.
Microsoft is not responsible for, nor does it grant any licenses to, third-party pack
Downloading https://github.com/protocolbuffers/protobuf/releases/download/v29.3/protoc-29.3-win64.exe
3.04 MB / 3.04 MB
Successfully verified installer hash
Extracting archive...
Successfully extracted archive
Starting package install...
Path environment variable modified; restart your shell to use the new value.
Command line alias added: "protoc"
Successfully installed
```



Check installed version

- In a [new terminal](#), check installation
- `protoc --version`

```
PS C:\Users\info> protoc --version  
libprotoc 29.3  
PS C:\Users\info> |
```

Protocol buffer compiler (**protoc**) — *is a compiler for protocol buffers definitions files(.proto files). It can generate C++, Java,golang,dart and Python source code for the classes defined in .proto file.*

Install dart compiler plugin



- Install the protocol compiler plugin for Dart (protoc-gen-dart)
- `dart pub global activate protoc_plugin`

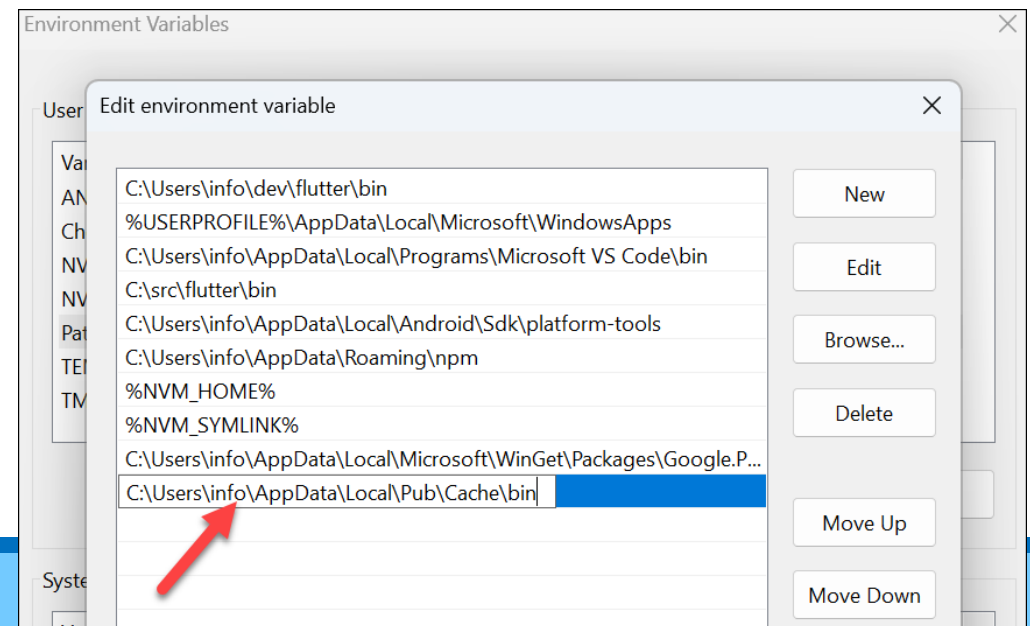
```
Windows PowerShell
PS C:\Users\info> dart pub global activate protoc_plugin

Downloading packages ... (1.1s)
+ collection 1.19.1
+ fixnum 1.1.1
+ meta 1.16.0
+ path 1.9.1
+ protobuf 3.1.0
+ protoc_plugin 21.1.2
Building package executables ... (1.7s)
Built protoc_plugin:protoc_plugin.
Built protoc_plugin:protoc_plugin_bazel.
Installed executable protoc-gen-dart.
Warning: Pub installs executables into C:\Users\info\AppData\Local\Pub\Cache\bin, which is not on your path.
You can fix that by adding that directory to your system's "Path" environment variable.
A web search for "configure windows path" will show you how.
Activated protoc_plugin 21.1.2.
PS C:\Users\info>
```

Update your path




- The plugin must be in your path!
- Add `C:\Users\<name>\AppData\Local\Pub\Cache\bin` to the user PATH variable
 - Or, update for YOUR path. See terminal for details.
- It generates Dart files for [working with data in protocol buffers format](#)



More info on the dart plugin




https://pub.dev/packages/protoc_plugin

 pub.dev

protoc_plugin 21.1.2

Published 17 months ago •  google.dev Dart 3 compatible

SDK DART FLUTTER PLATFORM ANDROID IOS LINUX MACOS WINDOWS

 88

[Readme](#) [Changelog](#) [Installing](#) [Versions](#) [Scores](#)

pub v21.1.2 publisher google.dev

This repository provides a Dart plugin for the [protoc compiler](#). It generates Dart files for working with data in protocol buffers format.

Requirements



Using the QuickStart App

- Follow the guidelines on grpc.io/docs/languages/dart/quickstart/ to start a text based Dart-version of server and client.
- Result:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\info> cd .\Desktop\grpc-dart\example\helloworld\
PS C:\Users\info\Desktop\grpc-dart\example\helloworld> dart .\bin\client.dart
Greeter client received: Hello, world!
PS C:\Users\info\Desktop\grpc-dart\example\helloworld> |
```

Quickstart
working

However, we want a Flutter implementation



- Create a new, basic Flutter app, or use existing app
 - Add `grpc` package: `flutter pub add grpc`
 - Add `protobuf` package: `flutter pub add protobuf`

```
flutter pub add grpc
Resolving dependencies...
Downloading packages... (2.1s)
+ args 2.7.0
+ async 2.12.0 (2.13.0 available)
+ crypto 3.0.6
+ fake_async 1.3.2 (1.3.3 available)
+ fixnum 1.1.1
+ google_identity_services_web 0.3.3
+ googleapis_auth 1.6.0
+ grpc 4.0.1
+ http2 2.3.1
+ leak_tracker 10.0.8 (10.0.9 available)
+ material_color_utilities 0.11.1 (0.12.0 available)
+ protobuf 3.1.0
+ vm_service 14.3.1 (15.0.0 available)
Changed 8 dependencies!
5 packages have newer versions available!
Try 'flutter pub outdated' for more information.

flutter pub add protobuf
Resolving dependencies...
Downloading packages...
+ async 2.12.0 (2.13.0 available)
+ fake_async 1.3.2 (1.3.3 available)
+ leak_tracker 10.0.8 (10.0.9 available)
+ material_color_utilities 0.11.1 (0.12.0 available)
+ protobuf 3.1.0 (from transitive dependency to direct dependency)
+ vm_service 14.3.1 (15.0.0 available)
```

Our example: `../examples/_550-gRPC-demo`

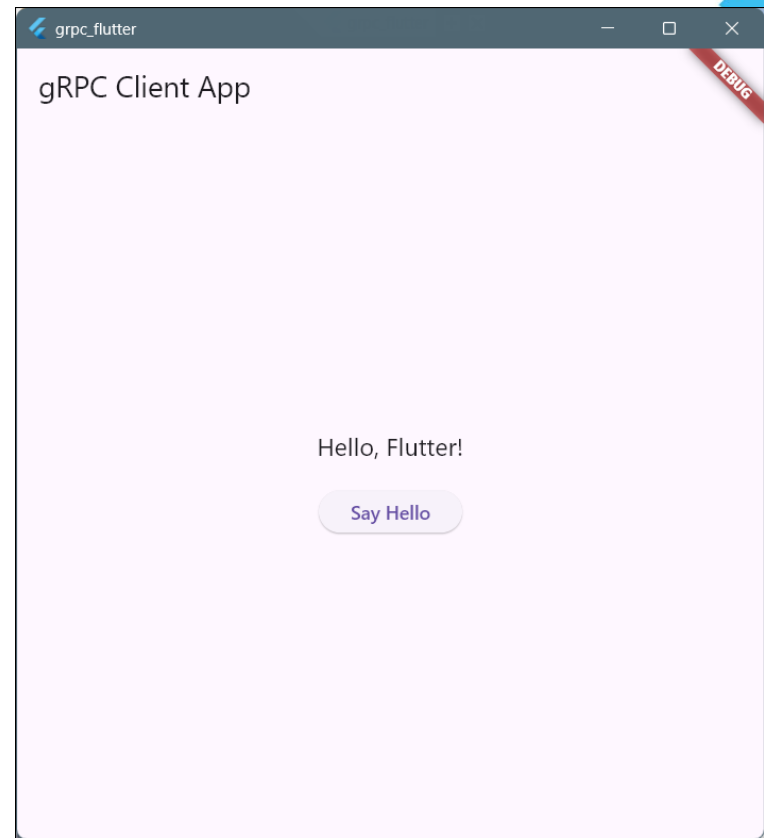
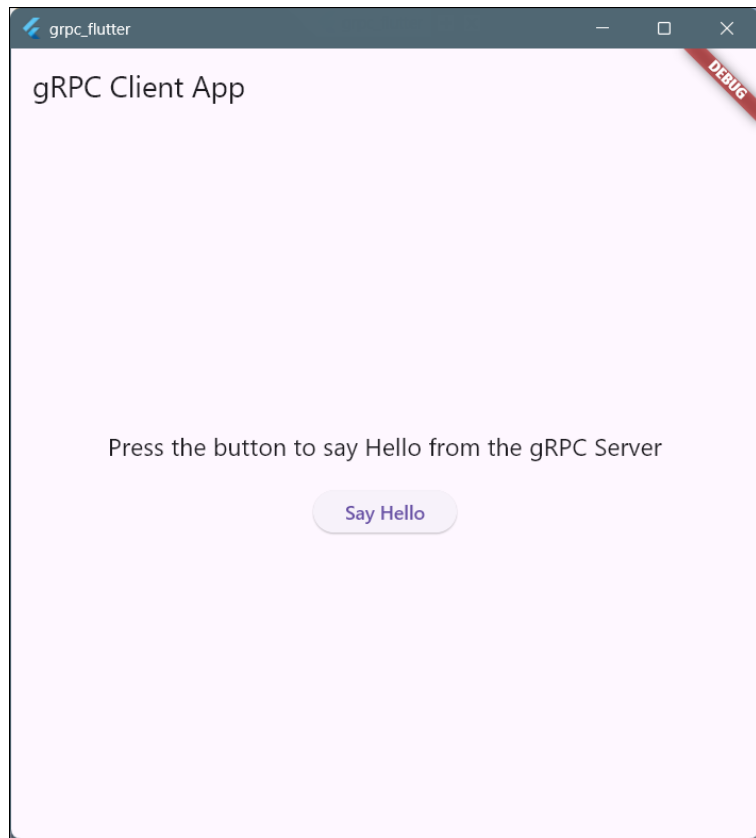


- Open the application
- In a terminal, `run the server`
 - `dart run ./server/server.dart`
- In another terminal, `run the client`
 - `flutter run`
- See `main.dart` for the homepage code

```
String _responseMessage = "Press the button to say Hello from the gRPC S

Future<void> _callGrpcService() async { PeterKassenaar, Today • added
    final channel = ClientChannel(
        'localhost', // Replace with your server's hostname or IP
        port: 50051,
        options: const ChannelOptions(
            credentials: ChannelCredentials.insecure(),
        ), // ChannelOptions
    ); // ClientChannel
```

Result





Optional: updating the app

- If you want to do something different, you have to
 - 1. update the server
 - 2. update the app
- For instance, in `helloworld.proto`, add a method

```
service Greeter {  
  // ...  
  // We now define a second method, that returns another greeting  
  rpc SayHelloAagain(HelloRequest) returns (HelloReply){  }  
}
```

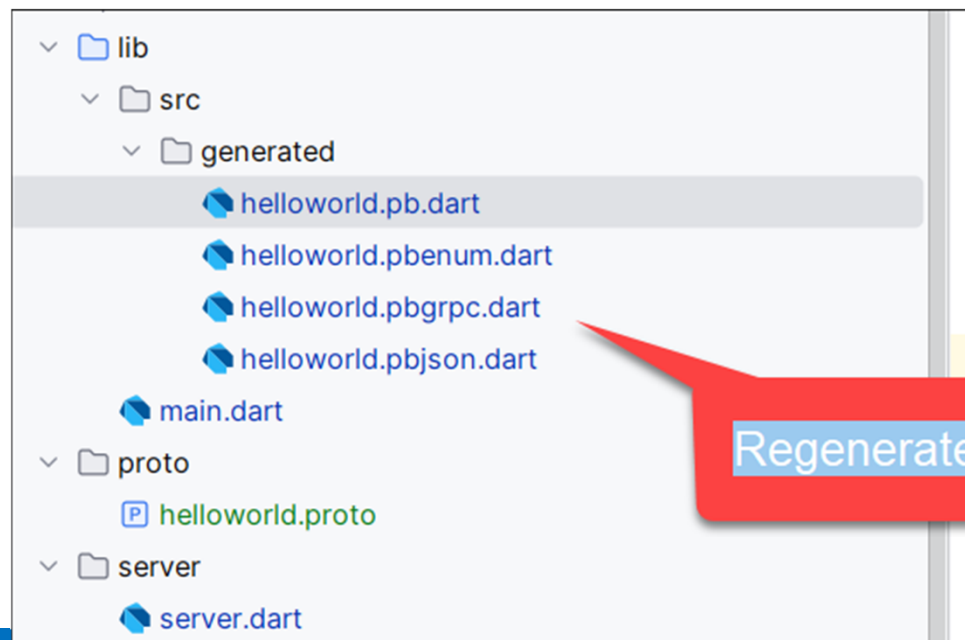
See `helloworld.proto` for other updates!

Regenerate gRPC Code



- Before we can use the new service method, we have to recompile the updated .proto file
- From the /proto directory, run

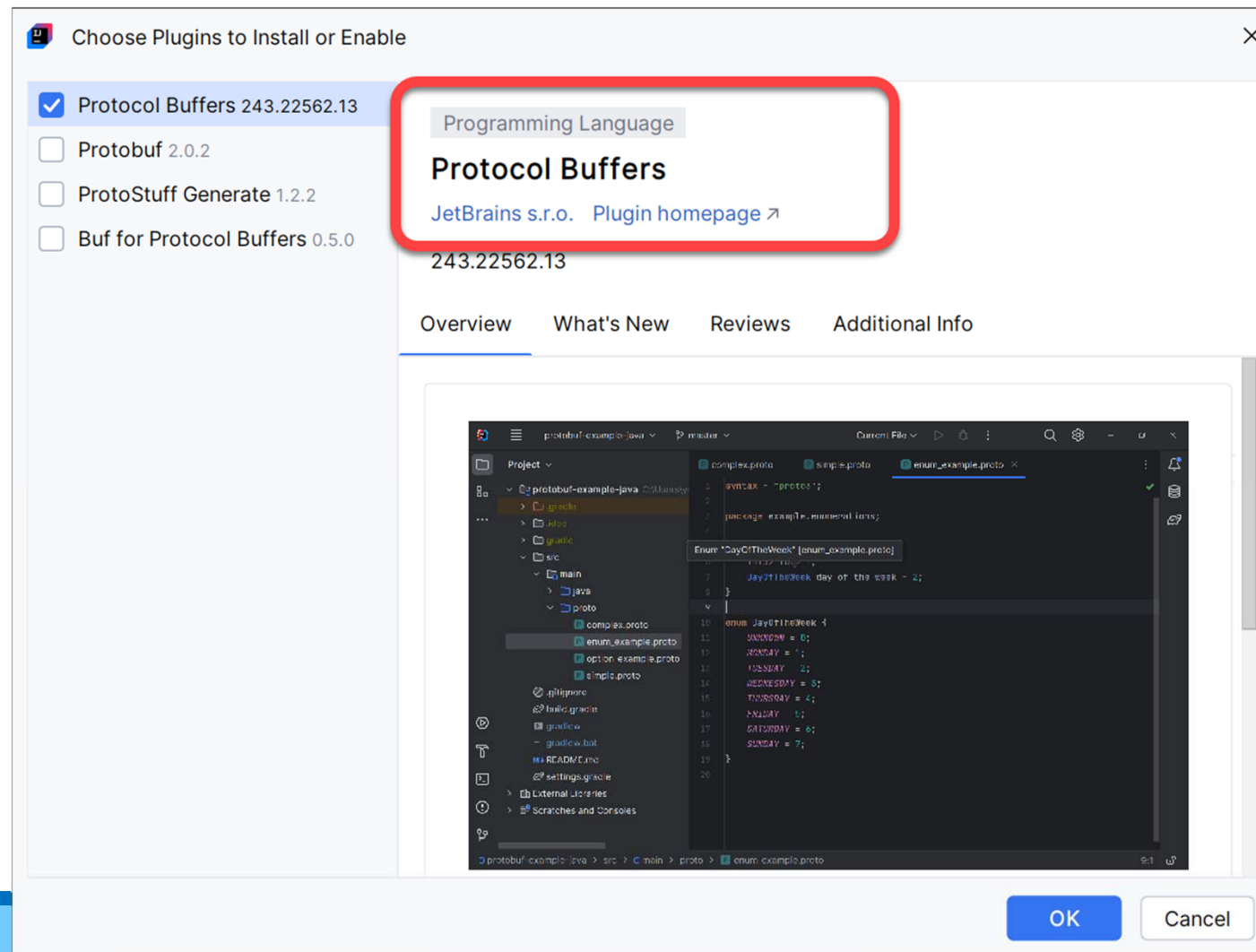
```
protoc helloworld.proto --dart_out=grpc:../lib/src/generated
```



Optional: .proto plugin



- IntelliJ can work with .proto files via a plugin



Update server.dart



- Update server.dart so the new method sayHelloAgain() can be called:

```
/// Dart implementation of the gRPC helloworld.Greeter server.  
class GreeterService extends GreeterServiceBase {  
  ...  
  @override  
  Future<HelloReply> sayHelloAgain(ServiceCall call, HelloRequest request,) async {  
    return HelloReply()..message = 'Hello again, ${request.name}!';  
  }  
}
```

Update main.dart



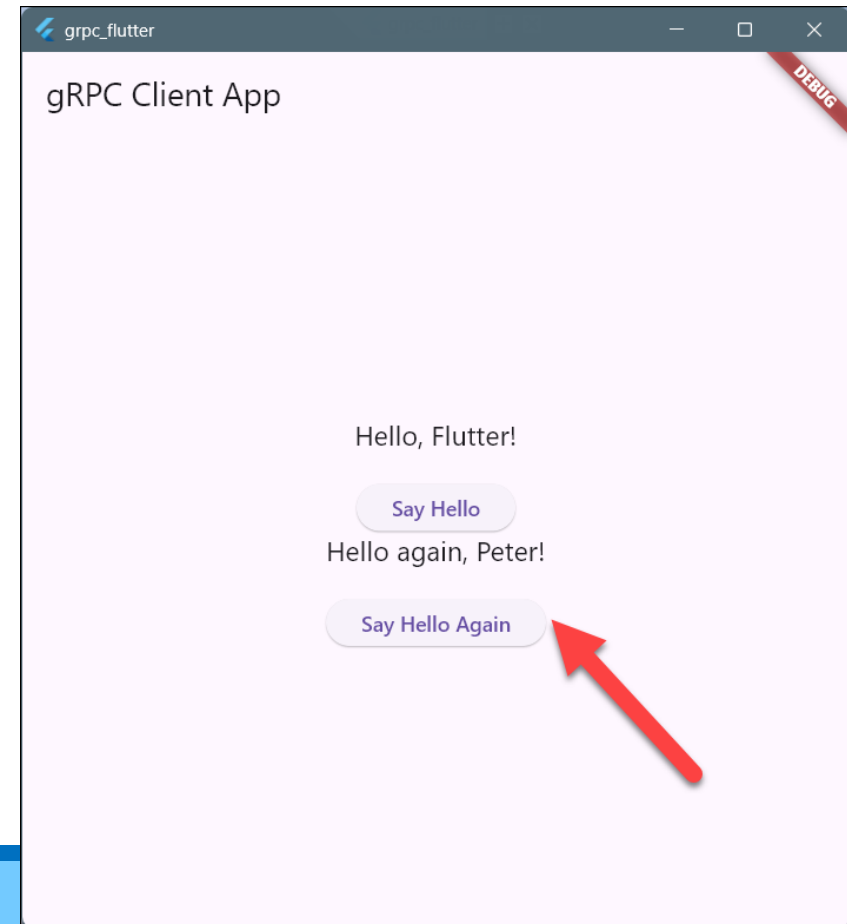
- Update main.dart so the new method is called
 - This one is simply duplicating the string. Of course you can adapt it to meet your needs.

```
...  
final response2 = await stub.sayHelloAgain(  
  HelloRequest()..name = 'Peter',  
)  
...  
// Create UI to call and show _response2...
```

Restart the server



- After updating and recompiling the server, **ALWAYS** stop a possible running instance (Ctrl+C) and restart the server
 - `dart run server.dart`
- Possible, simple UI:



Workshop



- Study the example on how to use gRPC server and client files
 - `../examples/_550-gRPC-demo`
- **Optional:** follow the Optional steps in this presentation to activate an extra method on the server
 - Extract `ClientChannel()` creation to its own function
- **Optional:** Create a completely new Flutter app, implementing gRPC Server and Client and get it working.
- Documentation
 - <https://grpc.io/docs/languages/dart/quickstart/>
 - <https://medium.com/@itachisasuke/using-grpc-with-flutter-5e1a8f553fa0>

