Get these slides ⇨

Introduction to the protobuf ecosystem

FOR: IBANFIRST R&D PAYMENT TEAM

Théophile Roos

orness.com

ORNESS

Contents

1 What is protobuf ?

Short for Protocol Buffers:

- an efficient, language-neutral data description language.
- a binary serialization format that use the language to describe the structure of the data.

Main features:

- Much smaller and faster than formats like JSON or XML
- Strong Typing and Schema: strict structure of the data defined in `.proto` files, enabling forward and backward compatibility
- Used for many use-cases: synchronous communication between network services, async communication through brokers systems (Kafka, RabbitMQ...), WebAssembly interfaces...

1.1 Syntax

Protobuf syntax versions

- `proto2`: legacy should not be used
- `proto3`: most used version currently
- `revision <year>`: new version with very limited support

First line of the `.proto` file

```proto
syntax = "proto3";
// syntax = "proto2";
// edition = "2023";
```

---

no syntax specified: proto2 assumed

1.1.1 Message definition

Message definition

```proto
message User {
  string name = 1 [json_name = "name"];
  optional int32 age = 2 [default = 18];

  // a comment
  repeated string hobbies = 3;

  map<string, int32> ibanToBalance = 4;
}
```

Fields are defined with:

- a FieldType: int32, string, bool, bytes, etc.
- a FieldName: name, age, etc.
- a Field Cardinality: optional, required, repeated, etc.

notes:

- *maps can't be repeated*

1.1.2 Enum definition

```proto
enum USER_TYPE {
  // a comment
  USER_TYPE_UNSPECIFIED = 0;
  USER_TYPE_REGULAR = 1;
  USER_TYPE_GUEST = 2;
  USER_TYPE_ADMIN = 3;
}

message User {
  // ...
  USER_TYPE user_type = 3;
}
```

The enum has a default value, which is the first value defined in the enum and with the FieldNumber `0`.

1.1.3 Enum definition (Aliases)

```proto
enum Corpus {
  option allow_alias = true;

  CORPUS_UNSPECIFIED = 0;
  CORPUS_UNIVERSAL = 1;
  CORPUS_WEB = 2;
  CORPUS_WWW = 2;
}
```

Message and enums can have options defined with the option keyword.

1.1.4 Message Composition

```proto
// user.proto
import "user_type.proto";

message User {
  // ...
  USER_TYPE user_type = 3;
}

message Organisation {
  string name = 1;
  repeated User users = 2;
}
```

```proto
// user_type.proto
enum USER_TYPE {
  // a comment
  USER_TYPE_UNSPECIFIED = 0;
  USER_TYPE_REGULAR = 1;
  // ...
}
```

An *Organisation* contains a list of *User* wich contain a field of type *USER_TYPE*

1.1.5 Nested Types

```proto
message User {
  enum USER_TYPE {
    USER_TYPE_UNSPECIFIED = 0;
    USER_TYPE_REGULAR = 1;
    // ...
  }
  // ...
  USER_TYPE user_type = 3;
}
```

Using nested type outside of the parent message definition

```proto
message UserProfile {
  User.USER_TYPE type = 1;
}
```

1.1.6 oneof

```proto
enum USER_TYPE {
  // ...
}
enum ADMIN_TYPE {
  // ...
}

message User {
  // ...
  oneof type {
    USER_TYPE user_type = 3;
    ADMIN_TYPE admin_type = 4;
  }
}
```

Using oneof to define a field that can be either a user or an admin.

Only one field of the oneof can be set at a time.

=> avoid using oneof if possible because of restrictions on the backward compatibility.

1.1.7 Packages

file structure:

```
./foo/bar/baz.proto
./fizz/buzz.proto
```

```proto
// foo/bar/baz.proto
package foo.bar;

message Baz {
  // ...
}
```

```proto
// fizz/buzz.proto
package fizz;

import "foo/bar/baz.proto";

message Buzz {
  foo.bar.Baz baz = 1;
}
```

1.1.8 Service definition

```proto
message GetUserRequest {
  string username = 1;
}
message GetUserResponse {
  User user = 1;
}

service UserService {
  rpc GetUser(GetUserRequest) returns (GetUserResponse);
  // rpc DeleteUser(DeleteUserRequest) returns (DeleteUserResponse);
}
```

A service *UserService* is defined with an RPC method *GetUser* that takes a *GetUserRequest* and returns a *GetUserResponse*.

1.1.8.1 What are RPCs ?

Remote Procedure Calls: a way to call a function on a remote server.

RPC based communication is a common pattern with many different implementations [1]:

- NFS: Network File System
- JSON-RPC: an RPC protocol that uses JSON-encoded messages.
- gRPC is a modern open source RPC framework that uses HTTP/2 and Protocol Buffers.
- IBM AIX: use of RPC in the AIX operating system. [2]

1.1.9 Option and Annotation

Protobuf has a way to add metadata to the messages and fields.

```proto
message User {
  string name = 1 [json_name = "name"];
}
```

The `json_name` field option is used to define the name of the field when it is serialized to JSON.

```proto
import "google/protobuf/descriptor.proto";
import "ibanfirst/base.proto"

extend google.protobuf.FieldOptions {
  string description  = 12345; // this must be unique
}
message User {
  option (ibanfirst.base.log).skip =true;
  string email = 1 [(description)= "This is the user's name"];
  string password = 2 [(description) = "This is the user's password"];
}
```

1.2 Usefull links

- Official documentation: https://protobuf.dev/overview/
- Un-official (but very good) documentation by the Buf team: https://protobuf.com/docs/introduction
- Third party protoc code generation plugins: https://github.com/protocolbuffers/protobuf/blob/main/docs/third_party.md
- Registed protobuf extension: https://github.com/protocolbuffers/protobuf/blob/main/docs/options.md#existing-registered-extensions
- somewhat standard google protobuf definitions: https://github.com/googleapis/googleapis

1.3 Protobuf based RPC

Some commonly used RPC framework

Framework	Transport Layer	Code Generation	Notable Features
gRPC	HTTP/2	Yes	Streaming, TLS, strong ecosystem
ConnectRPC	HTTP	Yes	Streaming, TLS, wide platform support
Ranger RPC	HTTP (any version)	Yes	Simple, fast, Go-centric
Apache Thrift	Any	Yes	Simple, fast, C++-centric
trpc	Any	No	Pure TypeScript

Side notes: gRPC can also use alternative serialization formats but it's uncommon.

1.4 gRPC

- gRPC: “gRPC Remote Procedure Calls”
- Started by Google in 2015, now a project of the Cloud Native Computing Foundation
- Transport layer: HTTP/2
- Interface Definition Language: Protocol Buffers
- Open-Source: Apache License 2.0
- Most used RPC framework
- Development by RFCs: https://github.com/grpc/proposal

Docs: https://grpc.io

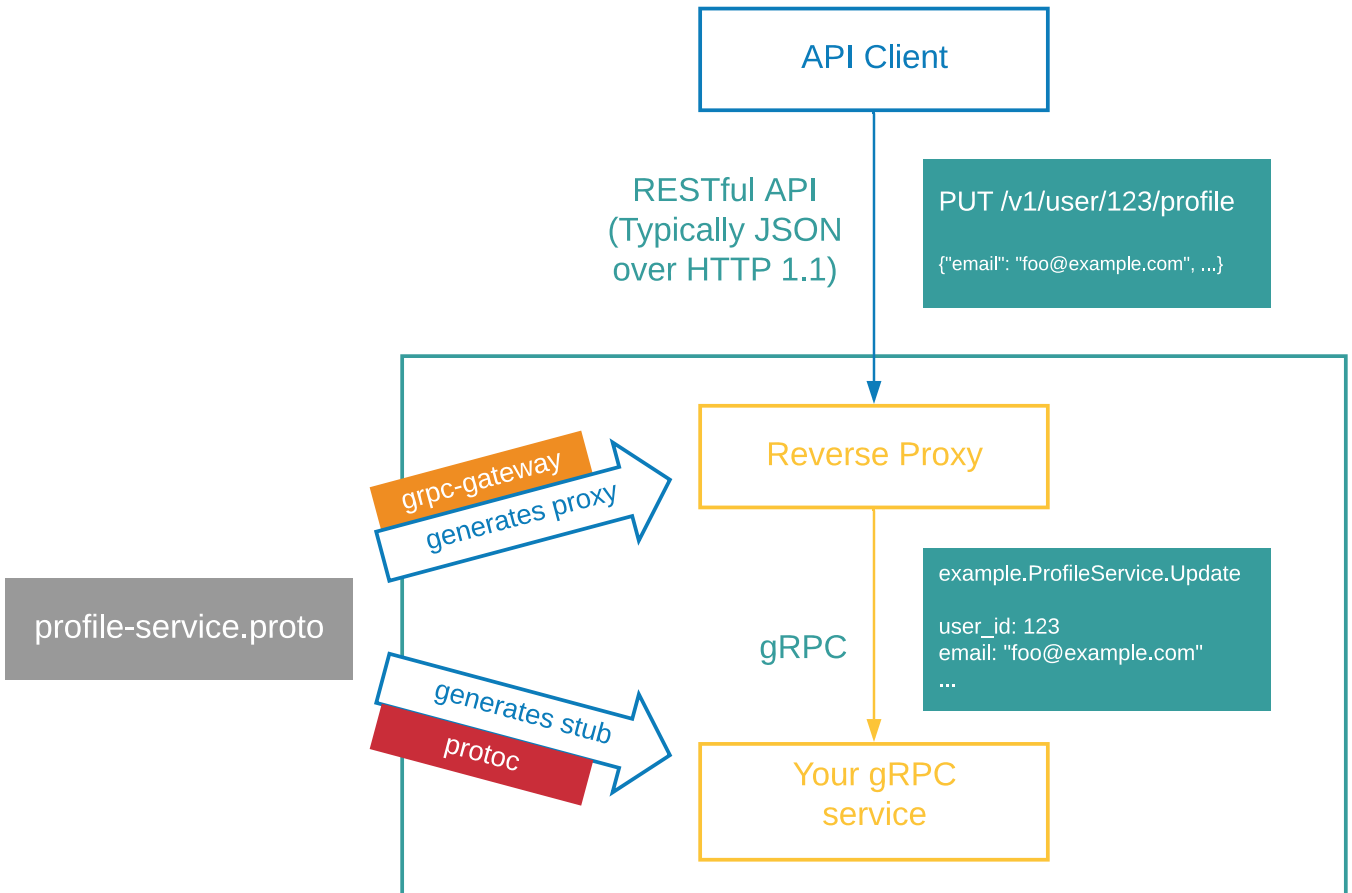
1.5 gRPC Features/Concepts

- Unary and Streaming RPC (client, server and bidirectional)
- Metadata (headers)
- Interceptors (middleware)
- Trailing metadata (HTTP/2 Trailers): used to send status codes, errors, etc.
- Deadlines/Timeouts: client communication timeouts to the server, which can know much time is left to respond.
- Cancellation: can be done by the client or the server, works on the server stops when the RPC is cancelled.
- Proxyless Load balancing: gRPC supports load balancing directly on the client side with the xDS protocol (xDS-based service discovery using a control plane)

1.6 gRPC-ecosystem

Suite of tools and libraries for gRPC

- *gRPC-gateway* : tool that converts gRPC APIs into RESTful APIs. Used to support legacy clients that are not capable of using gRPC.



- *openapiv2* : OpenAPI v2 specification generator for gRPC services.
- *grpc-opentracing* : OpenTracing instrumentation for gRPC.
- *grpc-health-probe* : gRPC health checking service.

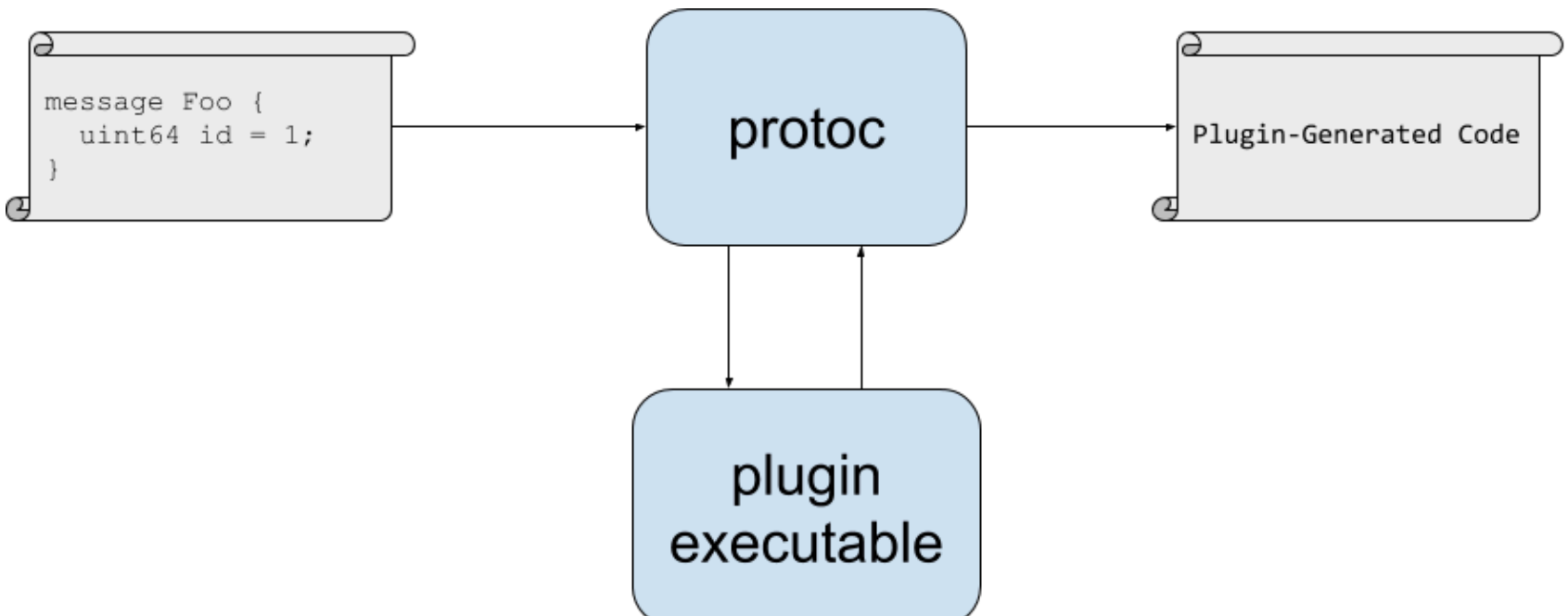
Docs: https://github.com/grpc-ecosystem/

2 Tools

- **protoc**: official compiler, usually used with makefiles.
  - pros: official implementation, very flexible (almost to much).
  - cons: somewhat slow, makefiles are not fun, no linting, mandatory vendoring.
- **prototool** [3]: deprecated but still used protobuf toolchain created by Uber.
  - pros: fast, good integration with IDEs, linting.
  - cons: no package management, deprecated and unmaintained.
- **buf**: new protobuf toolchain, created by buf.build [4].
  - pros: fast, good integration with IDEs, linting, package management.
  - cons: remote plugins send your code to their cloud

2.1 Official protobuf compiler: protoc

Open-Source Code: https://github.com/protocolbuffers/protobuf . Use plugins to generate code and artifacts from `.proto` files.



```shell
$ ls
./proto/foo/bar/baz.proto # our proto
./third_party/googleapi/annotation.proto
# vendored protos
```

```shell
protoc --proto_path=./protos \
  --proto_path=./third_party \
  --go_out=paths=source_relative:. \
  --go-grpc_out=paths=source_relative:. \
```

calls (*args passed via stdin*):

- protoc-gen-go
- protoc-gen-go-grpc

Proto files *are managed manually* (vendored proto code see [5])

2.2 buf.build cli: buf

All in one tool to manage your protobuf code using configuration first approach.

2.2.1 Migrating from protoc to buf

with protoc

Shell

```
protoc --proto_path=./protos \
  --proto_path=./third_party \
  --go_out=paths=source_relative:. \
  --go-grpc_out=paths=source_relative:. \
```

calls (args passed via stdin):

- protoc-gen-go
- protoc-gen-go-grpc

with buf

YAML

```yaml
#buf.yaml
version: v2
modules:
  - path: proto
deps:
  - buf.build/googleapis/googleapis
  - buf.build/bufbuild/protovalidate
lint:
  use: [ "STANDARD"]

# buf.gen.yaml
version: v2
plugins:
  - name: go # managed plugin
    out: .
    opt: paths=source_relative
  - name: go-grpc # managed plugin
    out: .
    opt: paths=source_relative
```

generate Go code with `buf generate`

References

[1] Contributors to Wikimedia projects, “Remote procedure call - Wikipedia.” [Online]. Available: https://en.wikipedia.org/w/index.php?title=Remote_procedure_call&oldid=1292581498

[2] “AIX.” [Online]. Available: https://www.ibm.com/docs/en/aix/7.3.0?topic=call-rpc-features

[3] “prototool.” [Online]. Available: https://github.com/uber/prototool

[4] “Buf - Crunchbase Company Profile.” [Online]. Available: https://www.crunchbase.com/organization/buf-technologies#people

[5] “Compilation and Descriptors \vert The Protobuf Language”. [Online]. Available: https://protobuf.com/docs/descriptors#standard-imports

[6] “Releases $c \cdot$ protocolbuffers/protobuf”. [Online]. Available: https://github.com/protocolbuffers/protobuf/releases/tag/v31.1

[7] “protobuf.” [Online]. Available: https://github.com/protocolbuffers/protobuf

[8] “Protocol Buffers.” [Online]. Available: https://protobuf.dev/

[9] “Introduction \vert The Protobuf Language”. [Online]. Available: https://protobuf.com/docs/introduction

[10] “Introduction - Buf Docs.” [Online]. Available: https://buf.build/docs/cli

3 That's it!

⇧Get these slides

Get in touch 👋

🌐 https://orness.com