1 - Course Introduction

The need for Protocol Buffers

An Evolution of data Comma Separated Values (CSV)

- Advantages:
    - Easy to parse
    - Easy to read
    - Easy to make sense of

- Disadvantages:
    - The data types of elements have to be inferred and is not a guarantee
    - Parsing becomes tricky when data contains commas
    - Column names may or may not be there

An Evolution of data Relational tables definitions

- Relational table definitions add types:

```
CREATE TABLE distributors (
  did   integer PRIMARY KEY,
  name  varchar(40)
);
```

- Advantages:
    - Data is fully typed
    - Data fits in a table
- Disadvantages:
    - Data has to be flat
    - Data is stored in a database, and data definition will be different for each database

An Evolution of data JSON (JavaScript Object Notation)

- JSON format can be shared across the network!

- Advantages:
    - Data can take any form (arrays, nested elements)
    - JSON is a widely accepted format on the web

  - JSON can be read by pretty much any language
  - JSON can be easily shared over a network

- Disadvantages:
  - Data has no schema enforcing
  - JSON Objects can be quite big in size because of repeated keys
  - No comments, metadata, documentation

An Evolution of data Protocol Buffers

```
syntax = "proto3";

message MyMessage {
  int32 id = 1;
  string first_name = 2;
  bool is_validated = 3;
}
```

- Protocol Buffers is defined by a .proto text file
- You can easily read it and understand it as a human

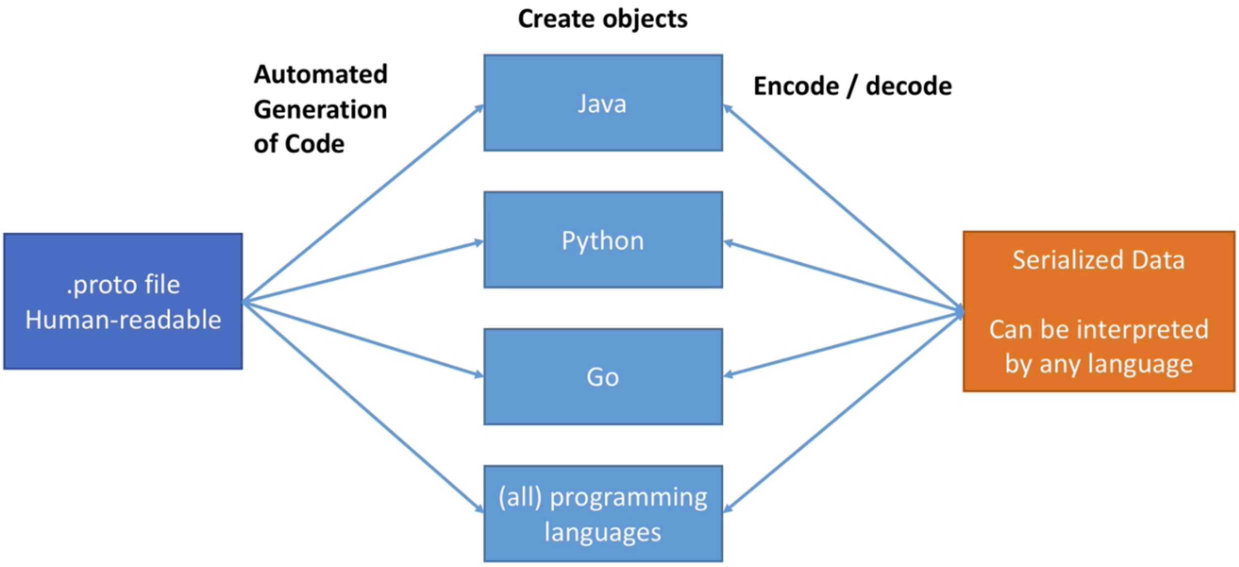
- Advantages
  - Data is fully typed
  - Data is compressed automatically (less CPU usage)
  - Schema (defined using .proto file) is needed to generate code and read the data
  - Documentation can be embedded in the schema
  - Data can be read across any language (C#, Java, Go, Python, JavaScript, etc…)
  - Schema can evolve over time, in a safe manner (schema evolution)
  - 3-10x smaller, 20-100x faster than XML
  - Code is generated for you automatically!

- Disadvantages:
  - Protobuf support for some languages might be lacking (but the main ones are fine)
  - Can't "open" the serialized data with a text editor (because it's compressed and serialised)

- Today Protocol Buffers is used as Google for almost all their internal applications.
- They have over 48000 Protobuf messages types in 12000 .proto files
- If it's working for Google, there's a great chance it'll be working for you!

How are Protocol Buffers used?

How is Protocol Buffer Used? To share data across languages!


Create objects
Automated Generation of Code
Encode / decode
.proto file
Human-readable
Java
Python
Go
(all) programming languages
Serialized Data
Can be interpreted by any language

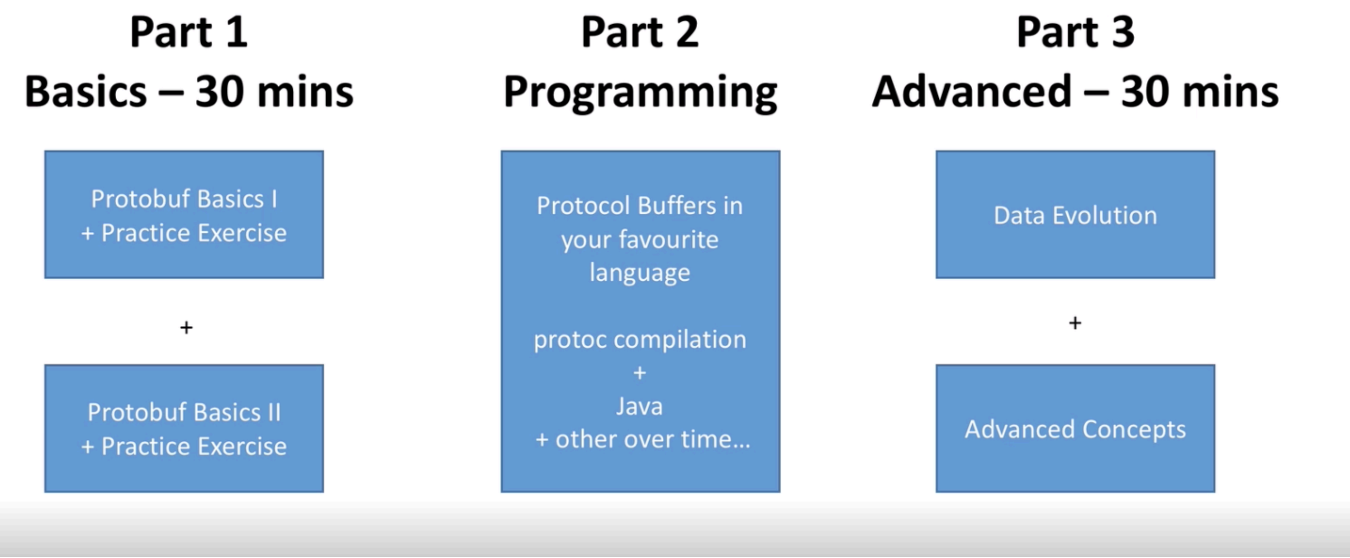
How is Protocol Buffer Used?

- Some databases may have support for Protocol Buffers data format.
- Lots of RPC frameworks, including gRPC, use Protocol Buffers to exchange data
- Google Uses it for all their internal API
- Some big projects like `etcd` use Protocol Buffers for transporting data

Proto2 vs Proto3

- Mid 2016, Google release the 3rd iteration of Protocol Buffers, named proto3
- We will only view proto3 in this course, as it will be the most common format used forward, and it has the best compatibility across a wide array of programming languages
- It is also the easiest to learn!

Course Structure



Courses Objectives

1. Write simple and complex .proto files
2. Practice Exercises to Confirm the learnings
3. Leverage Imports and Packages appropriately
4. Generate Code using `protoc` in any language
5. Code in Java/Python with Protocol Buffers
6. Understand how Data Evolution works for Protobuf
7. Leanr about advanced Protocol Buffers concepts

Pre-requisites

- Knowledge of one programming language is needed
- Previous experience with other formats such as XML or JSON is preferred
- Lots of willingness to learn something new!
- The course is made of short lectures to help you easily find content

Who is the course for?

- Developers who want to understand how to write .proto files and write code to create Protocol Buffer data
- Architect who want to understand how Protocol Buffers works and be useful for their solution architecture