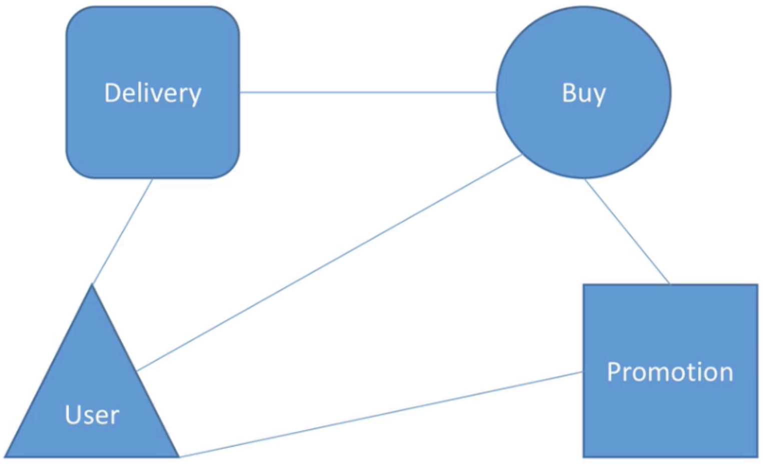
1 - gRPC Course Overview

gRPC Introduction

Today’s trend is to build microservices

- Microservices are built in different language nad encompass a function of your business:



- These microservices must exchange information and need to agree on:
  - The API to exchange data
  - The data format
  - The error patterns
  - Load Balancing
  - Many other
- One of the popular choice for building API is REST (HTTP-JSON)
- In this course, we’ll explore gRPC!

Building an API is hard

- Need to think about data model
  - JSON
  - XML
  - Something Binary?
- Need to think about the endpoint
  - GET /api/v1/user/123/post/456
  - POST /api/v1/user/123/post
- Need to think about how to invoke it and handle errors
  - API
  - Error

- Need to thihnk about efficiency of the API
    - How much data do I get out of one call?
    - Too much data
    - Too little data -> many API calls?
- How about latency?
- How about scalability to 1000s of clients?
- How about load balancing?
- How about inter operability with many languages?
- How about authentication, monitoring, logging?

Don’t you wish you could leave the boring and hard stuff to the framework?

What’s an API?

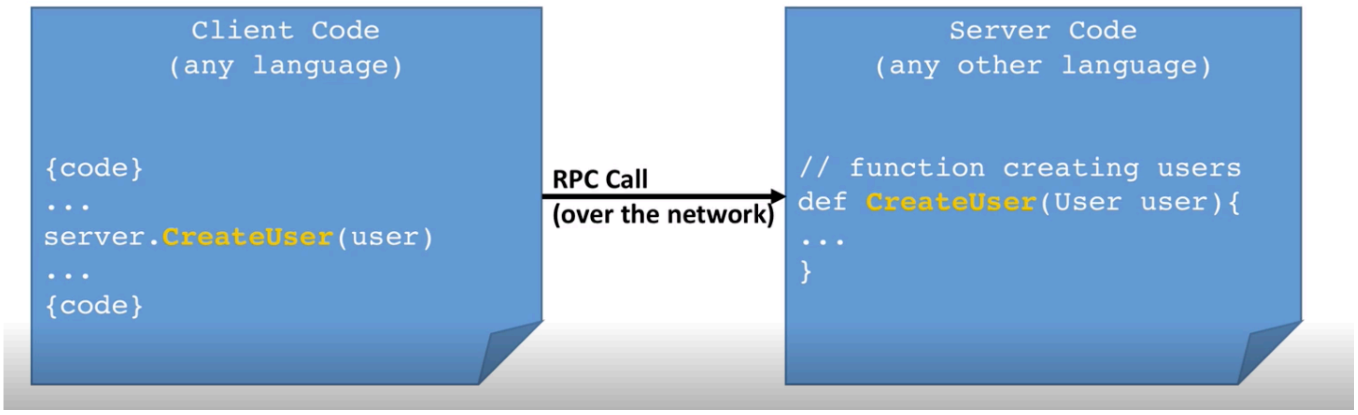
- At its core, an API is a contract, saying:
    - Send me this REQUEST (Client)
    - I’ll send you this RESPONSE (Server)
- It’s all about the data
- The rest, we’ll leave to the gRPC framework.

What’s gRPC?

- gRPC is a free and open-source framework developed by Google
- gRPC is part of the Cloud Native Computation Foundation (CNCF) - like Docker & Kubernetes for example
- At a high level, it allows you to define REQUEST and RESPONSE for RPC (Remote Procedure Calls) and handles all the rest for you
- On top of it, it’s modern, fast and efficient, build ohn top of HTTP/2, low latency, supports streaming, language independent, and makes it super easy to plug in authentication, load balancing, logging and monitoring.

What’s an RPC?

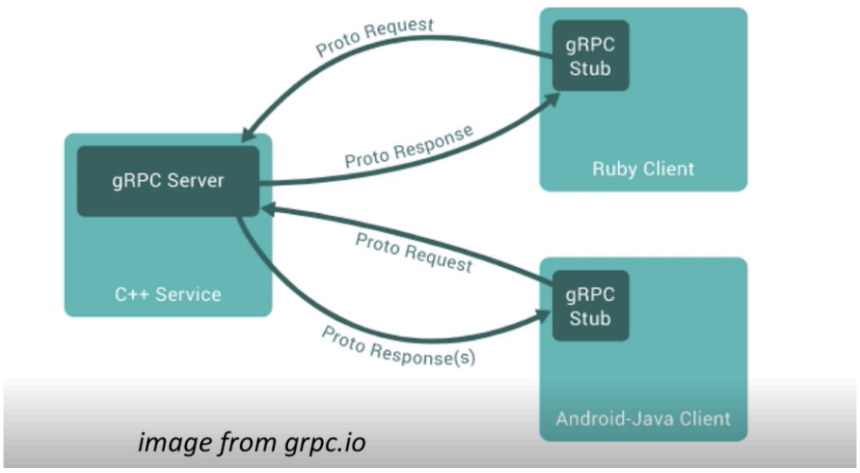
- An RPC is a Remote Procedure Call.
- In your CLIENT code, it looks like you’re just calling a function directly on the SERVER.



```
{code}
...
server.CreateUser(user)
...
{code}
```

```
// function creating users
def CreateUser(User user){
...
}
```

- It's not a new concept (CORBA had this before)
- With gRPC, it's implemented very cleanly and solves a lot of problems



How to get started?

- At the core of gRPC, you need to define the messages and services using **Protocol Buffers**
- The rest of the gRPC code will be generated for you and you'll have to provide an implementation for it.
- One **.proto** file works for over 12 programming languages (server and client), and allows you to use a framework that scales to millions of RPC per seconds.

Why Protocol Buffers?

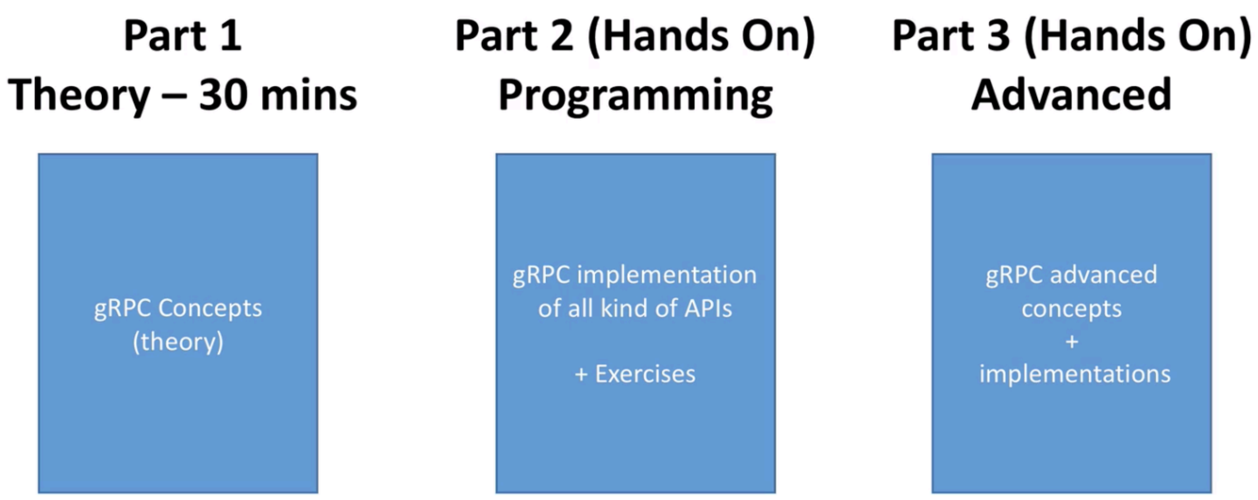
- Protocol Buffers are language agnostic
- Code can be generated for pretty much any language
- Data is binary and efficiently serialized (small payloads)
- Very convenient for transporting a lot of data
- Protocol Buffers allows for easy API evolution using rules

You should know the basics of Protocol Buffers before starting this course

Why should I learn it?

- Many companies have embraced it fully in Production
  - Google (internally and for Google Cloud services like Pub/Sub)
  - Netflix
  - Square (first contributor, replacement of all their APIs)
  - CoreOS (etcd 3 is built on gRPC for server-server communication)
  - Coackroach DB
- gRPC is the future of micro-services API and mobile-server API (and maybe Web APIs)

Course Structure



Course Objectives

1. Learn the gRPC theory to understand how gRPC works
2. Compare gRPC adn REST API paradigm
3. Write your gRPC service definition in .proto files
4. Generate Server & Client Code
5. Implememnt Unary, Server Streaming, Client Streaming & Bi-Directional Streaming API
6. Practice your learning with Exercises & Solutions
7. Implement advanced concepts such as Error Handling, Deadlines & SSL Security
8. Get pointers to expand your learning journey and get inspired by real world gRPC services

Pre-requisites

- Good Understanding of the Programming Language for this course
- Asynchronous programming is a plus
- Good Understanding of Protocol Buffers (see my protocol buffers course to get started)
- Lots of willingess to learn something new!
- This course will be challenging

Who is this course for?

- Developers who want to understand how to write gRPC Services and Clients
- Architects who want to understand how gRPC works and the concepts behind the different types of API