Introduction to gRPC

⇒ gRPC { Google's **Remote Procedure Calls** }

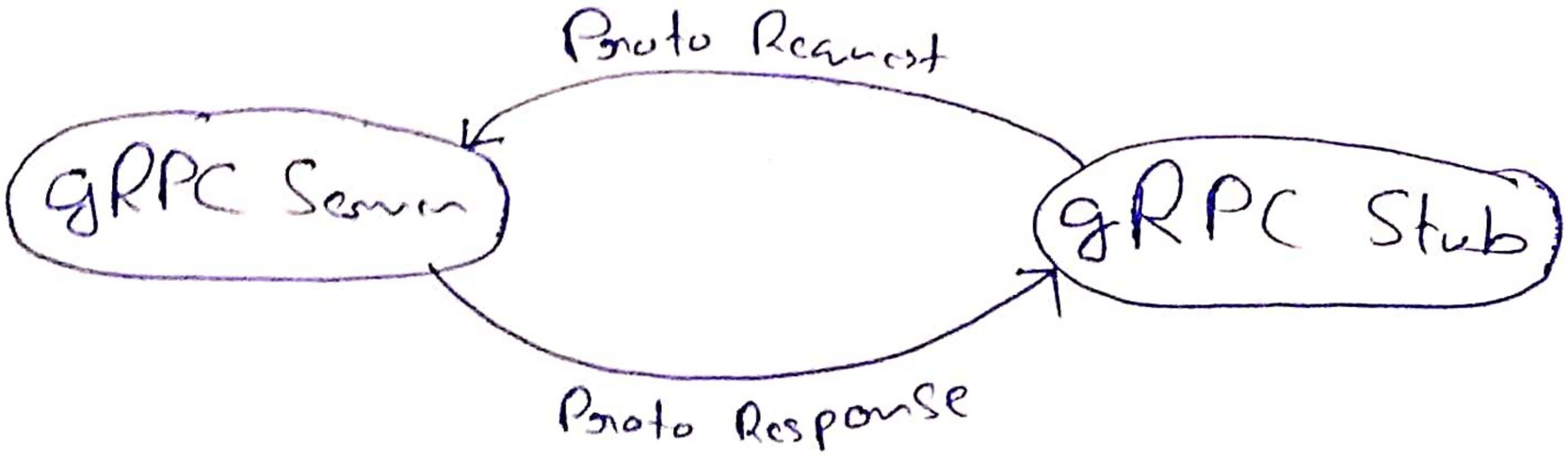
⇒ gRPC can use protocol buffers as both its:
- → Interface Definition Language (IDL)
- → and as its underlying message interchange format.

★ Overview

⇒ In gRPC, a client application can directly call a method on a server application on a different machine as if it were a local object, making it easier for you to create distributed applications and services.

⇒ gRPC is based around the idea of:
- → Defining a service
- → Specifying the methods that can be called remotely with their parameters & return types.

Server side	Client Side
⇒ Server implements the interface & runs a gRPC server to handle client calls.	⇒ Client has a Stub (referred to as just a client in some language) that provides the same methods as the server.



* Working with Protocol Buffers

⇒ By default, gRPC uses **Protocol Buffers**.

{ Google's open source mechanism for serializing structured data }

⇒ The first step when working with protocol buffers is to define the structure for the data you want to serialize in proto file.

↳ Ordinary text file with a **.proto** extension.

⇒ Protocol buffer data is structured as messages.

↳ Each message is a small logical record of information containing a series of name-value pairs called fields.

Example:

```
message Person {
    string name = 1;
    int32  id = 2;
    bool has_ponycopter = 3;
}
```

⇒ Then, you use the protocol buffer **compiler protoc** to generate data access class in your preferred language(s) from your proto definition.

→ These provide simple accessors for each field like name() and set_name(), as well as methods to serialize the whole structure to/from raw bytes.

⇒ You define gRPC services in ordinary proto files with RPC method parameters & return types specified as protocol buffer messages.

Example

```
Service Greeter{
    rpc SayHello (HelloRequest) return (Hello Reply){}
}

message HelloRequest{
    String name = 1;
}

message HelloReply{
    String message =1;
}
```

* Protocol buffer versions

⇒ While you can use proto2, but we recommend that you use proto3 with gRPC.