gRPC C++

⇒ On C++ world, there's no universally accepted Standard for managing project dependencies.

↳ You need to build & install gRPC.

★ Update the gRPC service

⇒ Update the application

↳ With extra method on the server for the client to call.

⇒ Recompile the updated proto

```
$ make -j2
```

⇒ This regenerates helloworld.pb.{h,cc} and helloworld.grpc.pb.{h,cc}, which contains the generated client & Server classes.

↳ As well as classes for populating Serializing and retrieving our request and response types.

Basic Tutorial

{Basic C++ programmer's introduction to working with gRPC}

⇒ With gRPC we can define our Service once in a .proto file and generate clients & servers in any of the gRPC's supported languages

route_guide.pb.h
↳ header which declares your generated message class.

route_guide.grpc.pb.h
↳ header which declares your generated service classes.

★ Creating the Server

⇒ There are two parts to making our RouteGuide Service do its job:

1) Implementing the service interface generated from our service definition
{Doing the actual work of our Service}

2) Running a gRPC server to listen for requests from clients & return the Service responses.

Context object (Point) {request} (Feature) {response}

=> Note: All Service methods can (& will) be called from multiple thread at the same time.
↳ You have to make sure that your method implementations are thread safe.

★ Starting the Server

=> Once we've implemented all our methods, we also need to start up a gRPC server so that clients can actually use our service.

1. Create an instance of our service implementation class RouteGuideImpl.
2. Create an instance of the factory ServerBuilder class.
3. Specify the address & port we want to use to listen for client requests using the builders AddListeningPort() method.
4. Register our Service implementation with the builder.
5. Call BuildAndStart() on the builder to create & start an RPC server for our Service.
6. Call wait() on the server to do a blocking until process is killed or shutdown() is called.

* Creating the Client

⇒ To call service methods, we first need to create a stub.

⇒ First we need to create a gRPC channel for our stub, specifying the server address and port we want to connect to.