Go: how to create a gRPC server

The world of cross-application request and response processing is constantly evolving, and one of the newest and most powerful approaches is the use of gRPC. gRPC is a Remote Procedure Call (RPC) framework developed by Google, which allows applications to communicate efficiently and reliably, regardless of the programming languages used. In this article, we will explore how to create a gRPC server using the Go programming language.

What is gRPC?

gRPC is a modern, high-performance RPC framework that uses the HTTP/2 protocol for client-server communication. It supports different types of programming languages and offers features such as defining services and messages using the Buffer Protocol (protobuf), two-way streaming, authentication, error handling and more. One of the main benefits of gRPC is its efficiency in handling communications, making it suitable for scenarios where performance is a priority.

Creating the server

Before you begin, make sure you have Go installed on your system. Next, youll need to install some dependencies needed to create a gRPC server. The main dependency is the Go gRPC package. You can install it using the following command:

```
go get -u google.golang.org/grpc
```

The next step is to define the service and messages that your gRPC server will handle. You can do this using `.proto` definition files. These files will define the operations of your service and the data that will be exchanged. For example, you might have an `example.proto` file:

```
syntax = "proto3";

package example;

service MyService {
  rpc GetData (DataRequest) returns (DataResponse);
}

message DataRequest {
  string request_param = 1;
}

message DataResponse {
  string response_data = 1;
}
```

Once the `.proto` files are defined, you will need to generate the corresponding Go codes using the protobuf compiler. Use the following command in the directory where your `.proto` file is located:

```
protoc --go_out=. --go-grpc_out=. example.proto
```

Now its time to implement the service you defined in the previous step. Open your Go file and import the packages generated by the protobuf compiler. Next, implement your service following the interface defined in the generated package.

```
package main

import (
    "context"
    "fmt"
    "net"

    "google.golang.org/grpc"
    "example"
)

type server struct{}

func (s *server) GetData(ctx context.Context, req
*example.DataRequest) (*example.DataResponse, error)
{
    response := &amp;example.DataResponse{
        ResponseData: "Hello, " + req.RequestParam,
    }
    return response, nil
}

func main() {
    listen, err := net.Listen("tcp", ":50051")
    if err != nil {
        log.Fatalf("Failed to listen: %v", err)
    }
    s := grpc.NewServer()
    example.RegisterMyServiceServer(s, &amp;server{})
    fmt.Println("Server listening on port 50051")
    if err := s.Serve(listen); err != nil {
        log.Fatalf("Failed to serve: %v", err)
    }
}
```

You can now start your gRPC server by running the command:

```
go run server.go
```