gRPC

This JavaScript code sets up a gRPC server and client. The server loads a protocol buffer file defining a `JokeService`, which generates a random joke from a JSON file whenever the `GetJoke` method is called. The client loads the same protocol buffer file and calls the `getJoke` method on the server, receiving and printing the random joke response. This code exemplifies a straightforward implementation of gRPC communication between a server and client in JavaScript.

Server.js

const grpc = require('@grpc/grpc-js');

const protoLoader = require('@grpc/proto-loader');

const packageDefinition = protoLoader.loadSync('joke.proto');

const jokePackage = grpc.loadPackageDefinition(packageDefinition);

const jokes = require('./jokes.json');

const server = new grpc.Server();

server.addService(jokePackage.JokeService.service, {

  GetJoke: (call, callback) => {

    const randomJoke = jokes[Math.floor(Math.random() \* jokes.length)];

    callback(null, { joke: randomJoke });

  }

});

server.bindAsync('0.0.0.0:3000', grpc.ServerCredentials.createInsecure(), () => {

});

{

  "dependencies": {

    "@grpc/grpc-js": "^1.10.1",

    "@grpc/proto-loader": "^0.7.10"

  }

}

Client.js

const grpc = require('@grpc/grpc-js');

const protoLoader = require('@grpc/proto-loader');

const packageDefinition = protoLoader.loadSync('joke.proto');

const jokePackage = grpc.loadPackageDefinition(packageDefinition);

const client = new jokePackage.JokeService(

    'localhost:3000',

    grpc.credentials.createInsecure()

);

client.getJoke({}, (error, response) => {

*if* (error) {

        console.error(error);

    } *else* {

        console.log(response);

    }

});

{

  "dependencies": {

    "@grpc/grpc-js": "^1.10.1",

    "@grpc/proto-loader": "^0.7.10"

  }

}

