gRPC

gRPC egy modern, nyílt forráskódú távoli eljáráshívást (Remote Procedure Call - RPC) megvalósító álltalános keretrendszer, amelyet a Google fejlesztett. Lehetővé teszi, hogy különböző platformokon és nyelveken írt alkalmazások egyszerűen kommunikáljanak egymással. gRPC protokollja alapértelmezetten Protocol Buffers-t (protobuf) használ.

Alapvető koncepciók:

- Szolgáltatások és metódusok: Egy .proto fájlban határozzuk meg a szolgáltatásokat, illetve azok távoli hívható metódusait.
- Kliens és szerver oldali kódgenerálás: Protobuf alapján automatikusan létrejönnek az interfészek és adattípusok.

Mintapélda

Hozzunk létre egy virtuális környezetet:

```
  python -m virtualenv ./venv
```

Majd aktiváljuk:

```
  ./venv/Scripts/activate
```

Telepítsük a függőségeket:

```
  python -m pip install grpcio
  python -m pip install grpcio-tools
```

Hozzuk létre a ./proto könvytárat és benne a IDL fájlt `helloworld.proto` néven:

```
syntax = "proto3";

// The greeting service definition.
service Greeter {
  // Sends a greeting
  rpc SayHello (HelloRequest) returns (HelloReply) {}
  // Sends another greeting
  rpc SayHelloAgain (HelloRequest) returns (HelloReply) {}
}

// The request message containing the user's name.
message HelloRequest {
  string name = 1;
}

// The response message containing the greetings
```

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

Futtassuk a stub generátort az alábbi paranccsal:

```
  python -m grpc_tools.protoc -I ./protos/ --grpc_python_out=. --
python_out=. .\protos\helloworld.proto
```

A helloworld_pb2.py és helloworld_pb2_grpc.py fájlok létrejönnek a gyökérben.

A kliens és a szerver kódja az alábbi lesz:

greeter_client.py

```
import grpc
import helloworld_pb2
import helloworld_pb2_grpc

def run():
    print("Will try to greet world ...")
    with grpc.insecure_channel("localhost:50051") as channel:
        stub = helloworld_pb2_grpc.GreeterStub(channel)
        response = stub.SayHello(helloworld_pb2.HelloRequest(name="you"))
    print("Greeter client received: " + response.message)
if __name__ == "__main__":
    run()
```

greeter_server.py

```
from concurrent import futures

import grpc
import helloworld_pb2
import helloworld_pb2_grpc


class Greeter(helloworld_pb2_grpc.GreeterServicer):
    def SayHello(self, request, context):
        return helloworld_pb2.HelloReply(message="Hello, %s!" %
request.name)

def serve():
    port = "50051"
    server = grpc.server(futures.ThreadPoolExecutor(max_workers=10))
    helloworld_pb2_grpc.add_GreeterServicer_to_server(Greeter(), server)
    server.add_insecure_port("[::]:" + port)
```

```
    server.start()
    print("Server started, listening on " + port)
    server.wait_for_termination()


if __name__ == "__main__":
    serve()
```

Indítsuk el a szervert és klienst.

Haladó lehetőségek

https://grpc.io/docs/guides/

- Authentication – Hitelesítési módszerek, beleértve saját mechanizmusok beépítését.
- Benchmarking – Teljesítménymérés eszközei és módszerei.
- Cancellation – RPC hívások megszakítása.
- Compression – Adatok tömörítése az átvitel előtt.
- Custom Backend Metrics – Saját metrikák gyűjtése szerver és kliens oldalon.
- Custom Load Balancing Policies – Egyedi terheléselosztási szabályok implementálása.
- Custom Name Resolution – Névfeloldás testreszabása.
- Deadlines – Határidők használata megbízhatatlan háttérszolgáltatások esetén.
- Debugging – Hibakeresés grpcdebug eszközzel.
- Error Handling – Hibakezelés és státuszkódok értelmezése.
- Flow Control – Manuális adatfolyam-vezérlés.
- Graceful Shutdown – Szerverek leállítása az ügyfelek megszakítása nélkül.
- Health Checking – Életjel támogatása szerver és kliens oldalon.
- Interceptors – Köztes rétegek alkalmazása általános funkcionalitásokhoz.
- Keepalive – HTTP/2 alapú kapcsolatébrentartás.
- Metadata – Kiegészítő adatok küldése fejlécekkel.
- OpenTelemetry Metrics – Megfigyelhetőség, mérőszámok gyűjtése.
- Performance Best Practices – Nyelvfüggő teljesítményoptimalizálási tanácsok.
- Reflection – Szolgáltatásleírások lekérdezése futásidőben.
- Request Hedging – Késleltetett kérések párhuzamos újraküldése.
- Retry – Újrapróbálkozás szabályozása részletesen.
- Service Config – Szolgáltatáskonfigurációs fájl kliens viselkedés befolyásolására.
- Status Codes – Részletes státuszkód lista és jelentések.
- Wait-for-Ready – Várakozás, míg a szerver készen áll a kiszolgálásra.

Feladatok

1. Próbáljuk ki a **Deadlines** módszert: hogyan lehet timeout-okat szabályosan kezelni?
2. Próbáljuk ki a **Reflection** módszert: hogyan lehet feltérképezni a szolgáltatás lehetőségeit?