Руководство по запуску gRPC на Ubuntu

- Создайте файл `service.proto` со следующим содержимым:

```
syntax = "proto3";

package example;
option go_package = "generated/;example";

service Greeter {
  rpc Add (Parm2Request) returns (Parm2Result);
  rpc Mul (Parm2Request) returns (Parm2Result);
  rpc Sub (Parm2Request) returns (Parm2Result);
  rpc Div (Parm2Request) returns (Parm2Result);
  rpc Pow2 (Parm1Request) returns (Parm1Result);
  rpc ReallyHeavyFunction (Parm2Request) returns (Parm1Result);
}

message Parm2Request {
  int32 x = 1;
  int32 y = 2;
}

message Parm1Request {
  int32 x = 1;
}

message Parm2Result {
  int32 x = 1;
  int32 y = 2;
  int32 z = 3;
}

message Parm1Result {
  int32 x = 1;
  int32 z = 2;
}
```

- Сервер
  - Создайте папку для сервера: `mkdir grpc-server && cd grpc-server`
  - Установите необходимые пакеты: `apt update && apt install -y golang-go protobuf-compiler ca-certificates && apt clean && rm -rf /var/lib/apt/lists/*`
  - Установите переменные окружения: `export GOPATH=/go && export`

`PATH=$PATH:$GOPATH/bin`

- Инициализируйте модуль Go: `go mod init go-grpc-server`
- Установите необходимые пакеты Go: `go get google.golang.org/grpc && go get google.golang.org/protobuf && go install google.golang.org/protobuf/cmd/protoc-gen-go@latest && go install google.golang.org/grpc/cmd/protoc-gen-go-grpc@latest`
- Скопируйте файл `service.proto` в папку: `cp ../service.proto service.proto`
- Сгенерируйте файлы-заглушки: `protoc --go_out=. --go-grpc_out=. service.proto`
- Создайте файл `server.go` со следующим содержимым:

```
package main

import (
    "context"
    "google.golang.org/grpc/codes"    "google.golang.org/grpc/status"
"log"    "net"    "time"
    pb "go-grpc-server/generated"
    "google.golang.org/grpc")

type server struct {
    pb.UnimplementedGreeterServer
}

func (s *server) Add(ctx context.Context, req *pb.Parm2Request)
(*pb.Parm2Result, error) {
    result := req.X + req.Y
    return &pb.Parm2Result{X: req.X, Y: req.Y, Z: result}, nil
}

func (s *server) Sub(ctx context.Context, req *pb.Parm2Request)
(*pb.Parm2Result, error) {
    result := req.X - req.Y
    return &pb.Parm2Result{X: req.X, Y: req.Y, Z: result}, nil
}

func (s *server) Mul(ctx context.Context, req *pb.Parm2Request)
(*pb.Parm2Result, error) {
    result := req.X * req.Y
    return &pb.Parm2Result{X: req.X, Y: req.Y, Z: result}, nil
}

func (s *server) Div(ctx context.Context, req *pb.Parm2Request)
(*pb.Parm2Result, error) {
    if req.Y == 0 {
       return nil, status.Errorf(codes.InvalidArgument, "division by
```

```
zero")
    }
    result := req.X / req.Y
    return &pb.Parm2Result{X: req.X, Y: req.Y, Z: result}, nil
}

func (s *server) Pow2(ctx context.Context, req *pb.Parm1Request)
(*pb.Parm1Result, error) {
    result := req.X * req.X
    return &pb.Parm1Result{X: req.X, Z: result}, nil
}

func (s *server) ReallyHeavyFunction(ctx context.Context, req
*pb.Parm2Request) (*pb.Parm1Result, error) {
    done := make(chan struct{})
    go func() {
       time.Sleep(time.Duration(req.X) * time.Second)
       close(done)
       log.Println("Function ended")
    }()
    select {
    case <-done:
       log.Println("Calculation is done")
       return &pb.Parm1Result{X: req.X, Z: 1337}, nil
    case <-ctx.Done():
       err := ctx.Err()
       log.Printf("Response will not be sent: %s", err)
       return nil, err
    }
}

func main() {
    lis, err := net.Listen("tcp", ":50051")
    if err != nil {
       log.Fatalf("failed to listen: %v", err)
    }
    s := grpc.NewServer()
    pb.RegisterGreeterServer(s, &server{})
    log.Println("Server is running on port :50051")
    if err := s.Serve(lis); err != nil {
       log.Fatalf("failed to serve: %v", err)
    }
}
```

- Запустите сервер: `go build -o grpc_server . && ./grpc_server`

```
root@aleh:/home/aleh/grpc-server# go build -o grpc_server . &&
./grpc_server
```

```
2024/12/05 13:48:42 Server is running on port :50051
```

- Клиент
  - Создайте папку для клиента: `mkdir grpc-client && cd grpc-client`
  - Установите необходимые пакеты: `apt update && apt install -y software-properties-common && add-apt-repository ppa:deadsnakes/ppa && apt update && apt install -y python3.12 python3.12-venv && apt clean && rm -rf /var/lib/apt/lists/*`
  - Скопируйте файл `service.proto` в папку клиента: `cp ../service.proto service.proto`
  - Создайте файл `main.py` со следующим содержимым:

```
import asyncio
import os

from grpc._channel import _InactiveRpcError
import grpc
from grpc.aio import AioRpcError

import service_pb2
import service_pb2_grpc


server_host = os.environ.get('GRPC_SERVER_HOST', 'localhost')
server_port = os.environ.get('GRPC_SERVER_PORT', '50051')


def run():
    with grpc.insecure_channel(f'{server_host}:{server_port}') as
channel:
        stub = service_pb2_grpc.GreeterStub(channel)
        response = stub.Add(service_pb2.Parm2Request(x=10, y=5))
        print(f"{response.x} + {response.y} = {response.z}")
        response = stub.Sub(service_pb2.Parm2Request(x=10, y=5))
        print(f"{response.x} - {response.y} = {response.z}")
        response = stub.Mul(service_pb2.Parm2Request(x=10, y=5))
        print(f"{response.x} * {response.y} = {response.z}")
        response = stub.Div(service_pb2.Parm2Request(x=10, y=5))
        print(f"{response.x} / {response.y} = {response.z}")
        try:
            x = 3
            print(f"Pow {x} ^ 2 = ", end="")
            response = stub.Pow2(service_pb2.Parm1Request(x=3),
timeout=3)
            print(response.z)
        except _InactiveRpcError as e:
            print(f"{e.details()}")
```

```
        try:
            print('Timeout example')
            response =
stub.ReallyHeavyFunction(service_pb2.Parm1Request(x=2), timeout=1.0)
            print(f"Result: {response.z}")
        except _InactiveRpcError as e:
            print(f"{e.args[0].code}: {e.details()}")


async def arun():
    async with grpc.aio.insecure_channel(f'{server_host}:
{server_port}') as channel:
        stub = service_pb2_grpc.GreeterStub(channel)
        response = await stub.Add(service_pb2.Parm2Request(x=45,
y=5))
        print(f"{response.x} + {response.y} = {response.z}")
        response = await stub.Sub(service_pb2.Parm2Request(x=12,
y=5))
        print(f"{response.x} - {response.y} = {response.z}")
        response = await stub.Mul(service_pb2.Parm2Request(x=5, y=5))
        print(f"{response.x} * {response.y} = {response.z}")
        response = await stub.Div(service_pb2.Parm2Request(x=76,
y=5))
        print(f"{response.x} / {response.y} = {response.z}")
        response = await stub.Pow2(service_pb2.Parm1Request(x=12))
        print(f"{response.x} ^ 2 = {response.z}")
        try:
            response = await stub.Div(service_pb2.Parm2Request(x=12,
y=0))
            print(f"Zero div: {response.z}")
        except AioRpcError as e:
            print(f'AIO RPC ERROR: {e.details()}')
        try:
            print("Cancellation example")
            future = stub.ReallyHeavyFunction(
                service_pb2.Parm1Request(x=2),
                # timeout=1.0,
                wait_for_ready=True
            )
            await asyncio.sleep(1)
            future.cancel()
            print(await future.details())
        except _InactiveRpcError as e:
            print(f"{e.args[0].code}: {e.details()}")

if __name__ == '__main__':
    print('Running sync')
    run()
    print('Running async')
    asyncio.run(arun())
```

- Создайте виртуальное окружение и установите нужные пакеты: `python3.12 -m venv .venv && source .venv/bin/activate && pip install grpcio grpcio-tools`
- Создайте файлы-заглушки: `python -m grpc_tools.protoc -I. --python_out=. --grpc_python_out=. --pyi_out=. service.proto`
- Запустите клиент: `python main.py`

```
(.venv) root@aleh:/home/aleh/grpc-client# python main.py
Running sync
10 + 5 = 15
10 - 5 = 5
10 * 5 = 50
10 / 5 = 2
Pow 3 ^ 2 = 9
Timeout example
StatusCode.DEADLINE_EXCEEDED: Deadline Exceeded
Running async
45 + 5 = 50
12 - 5 = 7
5 * 5 = 25
76 / 5 = 15
12 ^ 2 = 144
AIO RPC ERROR: division by zero
Cancellation example
Locally cancelled by application!
```