Руководство по запуску gRPC в Docker

1. Установить Docker на компьютер
    1. Для Windows
    2. Для Ubuntu
2. Создать на компьютере папку проекта со следующими файлами:

```
main.py
server.go
service.proto
client.Dockerfile
server.Dockerfile
```

3. Поместить в файл server.go следующий текст:

```
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)
    }
}
```

4. Поместить в файл service.proto следующий текст:

```
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;
}
```

5. Поместить в файл server.Dockerfile следующий текст:

```
FROM golang:1.23

RUN apt update
RUN apt install protobuf-compiler -y

ENV GOPATH=/go
ENV PATH=$PATH:$GOPATH/bin
WORKDIR /grpc_demo

RUN go mod init go-grpc-server
```

```
RUN go get google.golang.org/grpc
RUN go get google.golang.org/protobuf
RUN go install google.golang.org/protobuf/cmd/protoc-gen-go@latest
RUN go install google.golang.org/grpc/cmd/protoc-gen-go-grpc@latest

COPY service.proto server.go ./
RUN protoc --go_out=. --go-grpc_out=. service.proto
RUN go build -o grpc_server .
EXPOSE 50051
CMD ["./grpc_server"]
```

6. Поместить в файл client.Dockerfile следующий текст:

```
FROM python:3.12

WORKDIR /grpc-demo
COPY service.proto ./
RUN pip install grpcio grpcio-tools
RUN python -m grpc_tools.protoc -I. --python_out=. --grpc_python_out=. --
pyi_out=. service.proto
COPY client/main.py main.py

ENTRYPOINT ["python", "main.py"]
```

7. Поместить в файл main.py следующий текст:

```
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())
```

8. Выполнить команду `docker build -f server.Dockerfile -t grpc-server . && docker run -p 50051:50051 --name grpc-server grpc-server`
9. Выполнить команду `docker build -f client.Dockerfile -t grpc-client . && docker run --name grpc-client -e GRPC_SERVER_HOST=$(docker inspect -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' grpc-server) grpc-client`