python-grpc 开发与配置

- 环境：Linux centos7 python3.6
- 安装 python3.6 及 pip
- 安装所需 python 库
    - `pip3 install grpcio`
    - `pip3 install protobuf`
    - `pip3 install grpcio-tools`
- 编写 proto 文件，详见相关文档

  根据 proto 文件生成代码：python3 -m grpc_tools.protoc -I{相关 proto 文件目录} --python_out={python 输出位置} --grpc_python_out={grpcpython 输出位置} {proto 文件}
- 根据生成代码，完成功能
- 如果是服务端代码，需要新建一个 class，并继承上述步骤生成的 xxx_pb2_grpc.py 文件中的服务基类，然后从新定义服务方法：

```
import etcd_pb2_grpc as grpc

import etcd_pb2

from grpc import server

import time

from concurrent import futures


_ONE_DAY_IN_SECONDS = 60 * 60 * 24


class EtcdServer(grpc.GreeterServicer):

def Test(self, request, context):

str = request.name

return etcd_pb2.EtcdResponse(message="hello!%s" % str)


def serve():

grpcServer = grpc.server(futures.ThreadPoolExecutor(max_workers=4))
```

```
grpc.add_FormatDataServicer_to_server(EtcdServer(), grpcServer)
grpcServer.add_insecure_port("localhost:50051")
grpcServer.start()
try:
while True:
time.sleep(_ONE_DAY_IN_SECONDS)
except KeyboardInterrupt:
grpcServer.stop(0)

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

- 如果是客户端代码，直接按照下面方式调用即可

```
import etcd_pb2_grpc as grpc
import etcd_pb2
from grpc import insecure_channel

if __name__ == '__main__':
    conn = insecure_channel("localhost:50051")
    client = grpc.GreeterStub(channel=conn)
    request = etcd_pb2.EtcdRequest(name="wangtao")
    response = client.Test(request)
       print("received: " + response.message)
```

附录一：proto 文件

```
syntax = "proto3";

option java_outer_classname = "EtcdClient";
option java_package = "com.rouies.etcd";
option java_multiple_files = true;

package etcd;

service Greeter {
  rpc Test (EtcdRequest) returns (EtcdResponse) {}
```

```
}

message EtcdRequest {
  string name = 1;
}

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