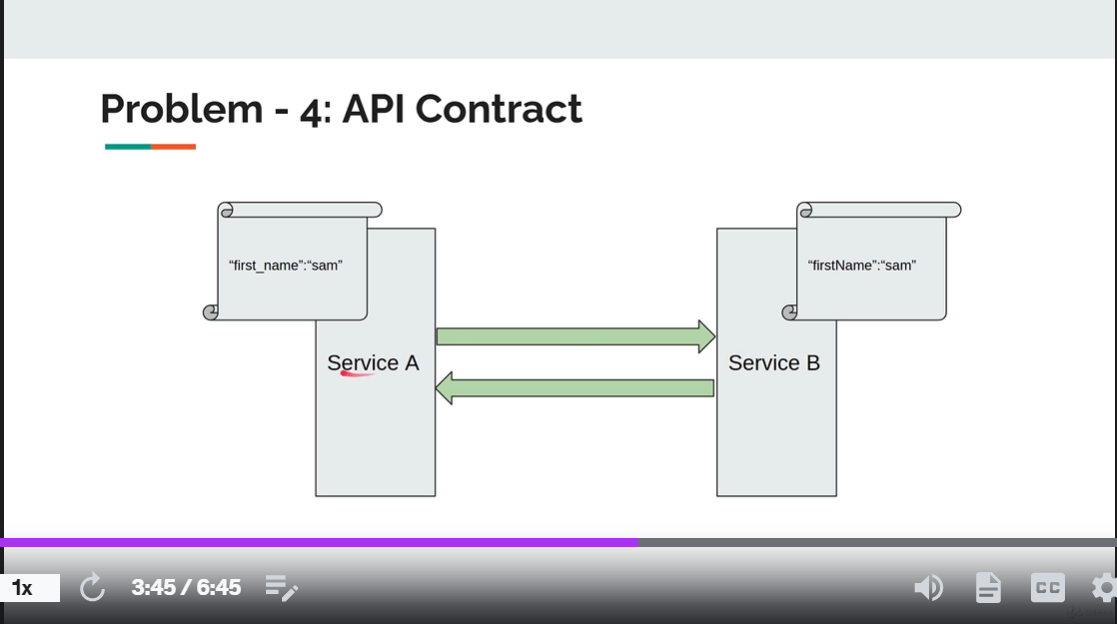
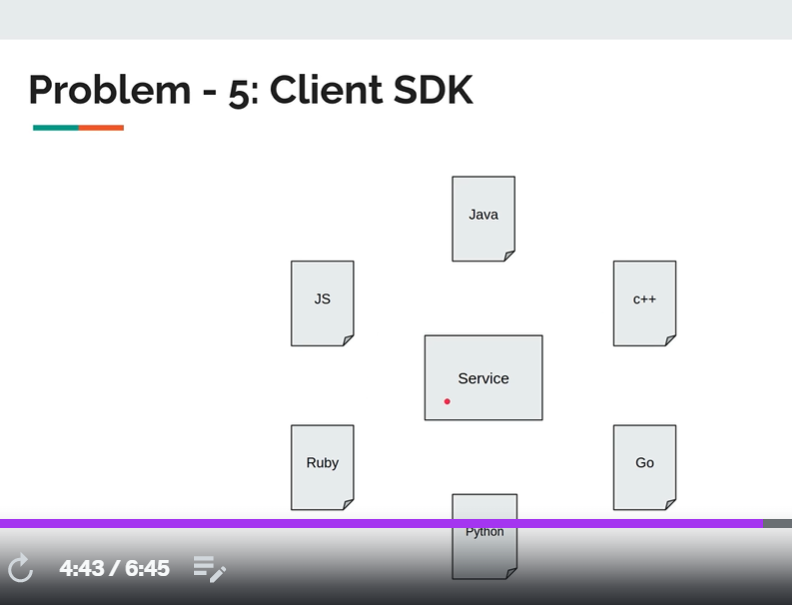
<https://github.com/grpc/grpc-java.git>

1. http disadvantages:

* http is stateless, so all information that is common between requests ( token, cookies) has to be shared many times in plain text and cannot be compressed => slow
* sending request body in json is human readable but it takes a lot of time to serialize and deserialize to bite so that computer can understand > take a lot of time
* 
* 
* => using http service cannot share it controller format with other client that not support http

1. Compare

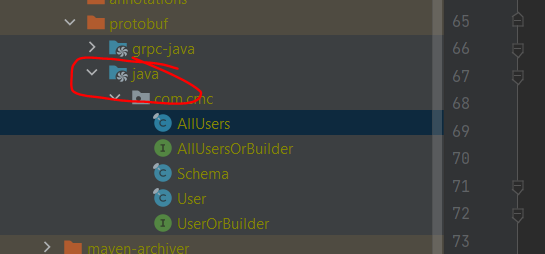
* Through put: request per second
* <https://www.baeldung.com/spring-rest-openapi-documentation>
* Default using http1 that has all above disadvantages

<https://www.baeldung.com/spring-rest-openapi-documentation>

1. Cannot resolve isStringEmpty => change plugin version:

<configuration>  
 <protocArtifact>com.google.protobuf:protoc:3.3.0:exe:${os.detected.classifier}</protocArtifact>  
 <pluginId>grpc-java</pluginId>  
 <pluginArtifact>io.grpc:protoc-gen-grpc-java:1.4.0:exe:${os.detected.classifier}</pluginArtifact>  
 <protoSourceRoot>${basedir}/src/main/proto/</protoSourceRoot>  
</configuration>

1. Install failed => make auto generated file to be



1. GRPC ⬄ REST

ProtoBuf ⬄ JSON

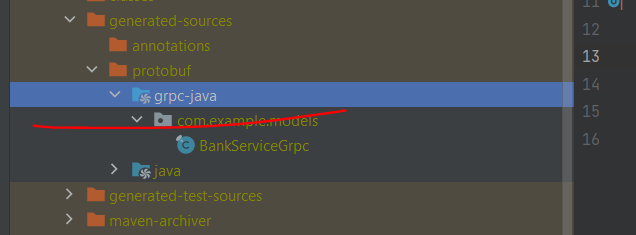
1. Rest using Json (Default format of dto in java) to serialize/ deserialize to send request => slwo

Grpc using protobuf to serialize/ deserialize to send request => fast

1. **Maybe => grpc is internal communication between microservices**

* Grpc is a framework not a protocol
* Grpc compare with rest: rest use http1, grpc use http2, payload rest use json, grpc use protobuf
* 2 type of communication between microservices: sync and async, sync: rest or grpc, async: kafka, message queue

1. Cannot find service impl base file => not in folder containing message, in grpc-service package



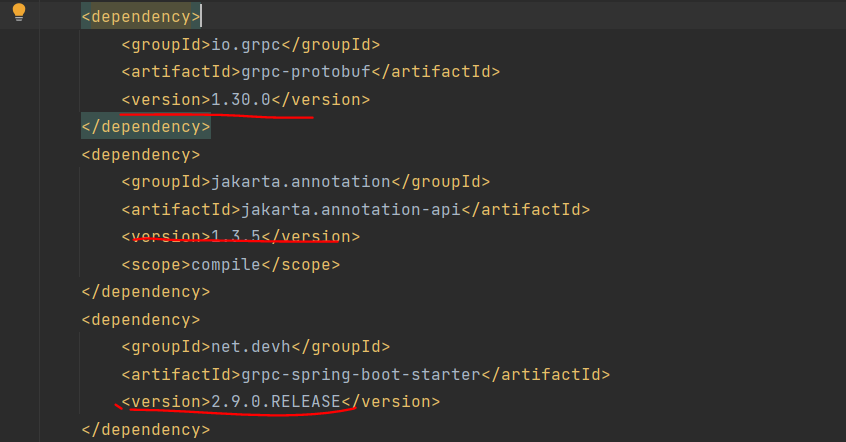
1. Response from server has 2 channel: data channel => on next

Error channel => on Error

1. Unary

Using blocking stub => waiting for response from server (all come together)

1. Pom problem => version

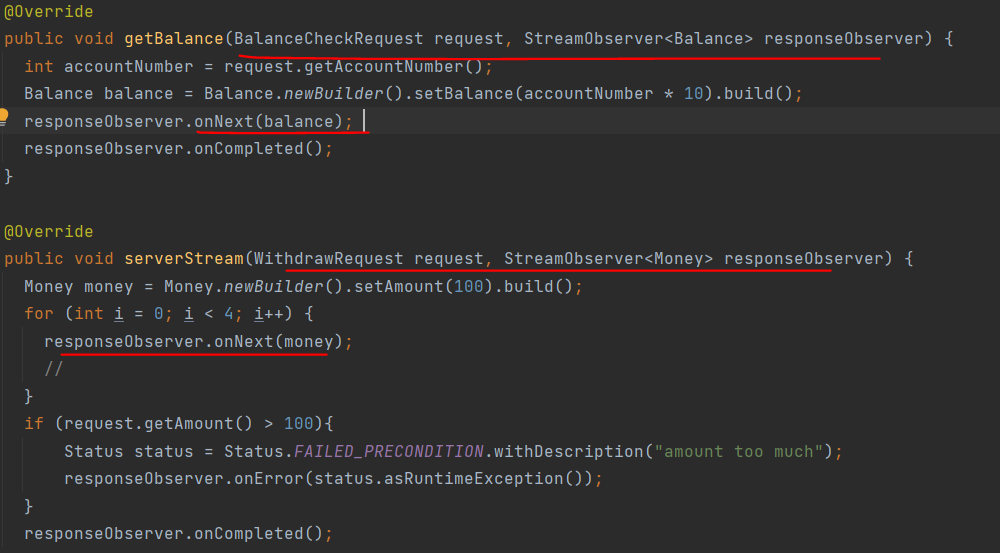


Don’t not add

<!-- <dependency>-->  
<!-- <groupId>io.grpc</groupId>-->  
<!-- <artifactId>grpc-netty-shaded</artifactId>-->  
<!-- <version>1.49.0</version>-->  
<!-- </dependency>-->

And javax dependency

If that, cannot compile generated file

1. 

Unary and server stream method signature are the same responseObServer in unary cannot have multiple onNext like server stream

1. From client side in server stream, client can use sync (blocking) to wait for all response from server or use async (stub) to continue without waiting response from server and *process each response from server*

* Faster compare with process everything from server and client do nothing

If client want to wait for calling server process to complete => use latch:

When server complete=> onComplete is call, in order to notify client that thread used to call server is done => use latch

1. Channel

A client want to connect with a server by grpc => must create a channel (ssl/tls connection )between those two, this channel will be used for all requests even concurrent request => don’t shutdown the channel except when server is down

5. os.detecked version cannot resolve => make sure all version is added to all dependency