Protocol buffer through Kafka Python Producer and Java Consumer

samrat.saha Follow
Nov 25, 2019 · 3 min read

protobuf through mixed Kafka producer and consumer

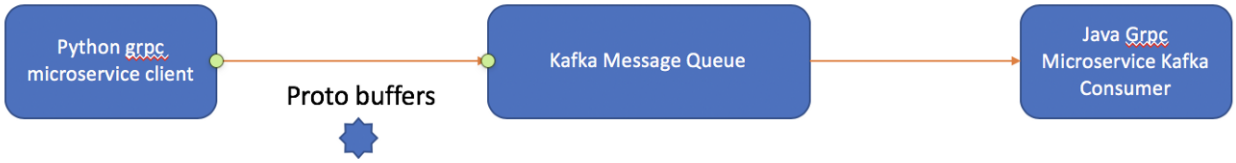
JAVA and Python GRPC Micro services communication through Kafka messaging:

In this post, we discuss one of the most common issues faced while building a microservice architecture using python and java over grpc and protocol buffers.

Passing a protocol buffer from a python client and consuming in a grpc JAVA based microservice is problematic due to byte difference in Java and Python.

Just for bulk scale message processing we will also have Kafka Message queue in between.

Have a look at the below generic architecture which we will see how to code.



Python and Java protocol buffer message passing through kafka

Versions:

- Java — 1.8
- python 3.6
- protoc 3.10.1
- pykafka

<groupId>org.apache.kafka</groupId>

<artifactId>kafka-clients</artifactId>

<version>2.0.0</version>

> *Lets Define a Simple **Protocol buffer** message for communication*

```
syntax = "proto3";

//import "google/protobuf/empty.proto";
//import "google/protobuf/timestamp.proto";

option java_package = "com.kafka.simple.genproto";
import "google/protobuf/empty.proto";


message Simple {
    string label = 1;
    string description = 2;
    int32 version = 3;
}
```

> *For Grpc microservice just implement a grpc service like below*

```
service SimpleMessageService {
    rpc pushSimpleMessage(Simple) returns (google.protobuf.Empty);
}
```

> *Python Kafka Client to populate our Simple Message (pykafka)*

Base64 encoding here is really important as simple SerilizeToString wont work on the Java consumer side.

```
client = KafkaClient(hosts="127.0.0.1:9092")

topic = client.topics["simpletopic"]

msg = simple_pb2.Simple()
msg.label = 'simple message'
msg.description = 'this is a simple message'
msg.version = 1
print(msg)

with topic.get_sync_producer(max_retries=5, required_acks=1,
ack_timeout_ms=5000) as producer:
    #this is importontant to base64 encode the message
    #just using SerilizeToString wont work on the other end java
consumer
    b64encoded = base64.b64encode(msg.SerializeToString())
    res = producer.produce(b64encoded)
    print(res)
```

> *Kafka-Console-Consumer how does it look*

```
kafka-console-consumer.sh - bootstrap-server 127.0.0.1:9092 - topic
"simpletopic"
Cg5zaW1wbGUgbWVzc2FnZRIYdGhpcyBpcyBhIHNpbXBsZSBtZXNzYWdlGAE=

Base64 Encoded protocol buffer
```

> *Java Kafka Consumer thread (only the important decoding part)*

Check the Base64 decoding portion of the code.

```
public void run()
{
    logger.info("consumer thread [" + this.id + "] starting up");
try {
consumer.subscribe(topics);

while (true) {
ConsumerRecords < String, String > records =
consumer.poll(Duration.ofMillis(100));
for (ConsumerRecord < String, String > record: records) {
try {
logger.info("consumer thread [" + this.id + "] received \n" +
record);
Simple msg =
SimpleOuterClass.Simple.parseFrom(Base64.getDecoder().decode(record.
value()));

logger.info("consumer thread [" + this.id + "] received \n" +
msg.toString());
// Push to KGIngestionService MicroService

consumer.commitAsync(new OffsetCommitCallback() {
public void onComplete(Map < TopicPartition, OffsetAndMetadata >
offsets, Exception exception) {
if (exception != null)
logger.info("Commit failed for offsets {" + offsets + "  }" + "
exception " +
exception);
else
logger.info("Commit for offsets {" + offsets + "  } successful");
}
});

}
```

Hope you got the idea how to pass protocol buffer messages through python Kafka client and java Kafka consumer.

This should work either way only Base64 Encoding and Decoding part has to change.

Hope this will help you build scalable microservices architecture using Grpc, Protocol Buffer and Kafka.

Thank you.

References:

[1] https://pykafka.readthedocs.io/en/latest/

[2] https://developers.google.com/protocol-buffers

[3] https://kafka.apache.org/10/javadoc/index.html?org/apache/kafka/clients/consumer/KafkaConsumer.html

Protocol Buffers Grpc Kafka Python Java