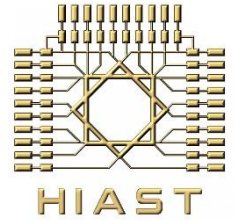


Syrian Arab Republic

Higher Institute for Applied Science and Technology

Fifth Year



Distributed Systems

By student: Hussein Salloum

2025-2026

1. ما هو بروتوكول (Protobuf)؟

بروتوكول (Protocol Buffers)، أو اختصاراً Protobuf، هو آلية لترميز البيانات المربطة (serialization) تم تطويرها بواسطة google. يُستخدم لتحويل البيانات إلى تنسيق ثنائي (binary) فعال ومحايد للغة البرمجة، مما يسمح بنقل البيانات بين أنظمة مختلفة بكفاءة عالية. يركز Protobuf على جعل البيانات أصغر حجماً وأسرع في التحليل مقارنة بتنسيقات أخرى مثل JSON أو XML، وهو يدعم التوافق الخلفي (backward compatibility) لتعديل الهيكل دون كسر التطبيقات القائمة.

2. ما هي الميزتان الرئيسيتان لاستخدام gRPC؟

gRPC هو إطار عمل للاتصال عن بعد (Remote Procedure Call - RPC) يعتمد على HTTP/2 ويستخدم Protobuf كتتنسيق افتراضي للبيانات. الميزتان الرئيسيتان هما:

- الأداء العالي: يدعم التدفق الثنائي (bidirectional streaming)، الضغط التلقائي، والتعددية (multiplexing) عبر HTTP/2، مما يجعله أسرع وأقل استهلاكاً للموارد مقارنة بالبدائل التقليدية.
- الدعم لعدة لغات برمجة: يولد gRPC كوداً تلقائياً للعميل والخادم في لغات متعددة مثل Java، Python، Go، وC++، مما يسهل التكامل بين أنظمة متنوعة مع ضمان التوافق.

3. وضح الفروق بين RESTful و gRPC متى يمكننا أن نفضل استخدام أحدهما عن الأخرى

الفروق الرئيسية:

- البروتوكول الأساسي: RESTful APIs تعتمد عادةً على HTTP/1.1 مع أفعال مثل GET، POST، PUT، DELETE، بينما gRPC يعتمد على HTTP/2 لدعم التدفقات المتعددة والثنائية.
- تنسيق البيانات: REST غالباً ما يستخدم JSON أو XML (نصي وسهل القراءة)، أما gRPC فيستخدم Protobuf (ثنائي وأكثر كفاءة في الحجم والسرعة).
- النمط: REST هو نمط معماري stateless يركز على الموارد (resources)، بينما gRPC هو نمط RPC يركز على الإجراءات (procedures) مع دعم للتدفق (streaming) مثل client-streaming أو server-streaming.
- الأداء والكفاءة: gRPC أفضل في السيناريوهات ذات الحمل العالي أو الاتصالات الداخلية بسبب الضغط والتعددية، بينما REST أبسط وأكثر انتشاراً للواجهات العامة.
- الأمان والميزات: كلاهما يدعم TLS، لكن gRPC يدعم المصادقة التلقائية والتدفقات الطويلة الأمد بشكل أفضل.

متى نفضل أحدهما:

- فضل RESTful: عند بناء APIs عامة (public-facing) تحتاج إلى سهولة الاستخدام، التوافق مع المتصفحات، أو التكامل مع أدوات موجودة مثل Swagger. مثال: تطبيقات الويب أو خدمات خارجية حيث تكون القراءة البشرية مهمة (أي أهم فكرة أنه نستخدمه عند عدم وجود اتفاقات بين الزبون والمخدم)

- فضل gRPC: في الخدمات الداخلية (microservices)، التطبيقات ذات الأداء العالي مثل IoT أو الذكاء الاصطناعي، أو عند الحاجة إلى تدفق بيانات مستمر (مثل الدردشة أو البث). يُفضل أيضاً في البيئات متعددة اللغات حيث يقلل من وقت التطوير.

4. ما هو الغرض من ملف proto. في سياق Protocol Buffers؟

ملف proto. هو ملف تعريف نصي يحدد هيكل البيانات في Protobuf. غرضه الرئيسي هو وصف الرسائل (messages) والخدمات (services) بشكل رسمي، بما في ذلك الحقول، الأنواع، والعلاقات. عند تجميعه (compilation) باستخدام أداة protoc، يولد كوداً مصدرياً (source code) للغات برمجة مختلفة، مما يسمح بترميز وفك ترميز البيانات بكفاءة. هذا يضمن التوافق عبر الإصدارات ويسهل التكامل بين الأنظمة.

1. ما هو دور المترجم protoc في مشروع gRPC؟

المترجم protoc (اختصار لـ Protocol Compiler) هو أداة أساسية في مشاريع gRPC. دوره الرئيسي هو تحويل ملفات التعريف proto. إلى كود مصدري (generated code) بلغات البرمجة المختلفة مثل Java، Python، Go، أو ++C. هذا الكود يشمل تعريفات الرسائل (messages) والخدمات (services)، بالإضافة إلى الكود الخاص بـ gRPC للعميل (client) والخادم (server). يساعد protoc في ضمان التوافق بين الأنظمة، تسهيل عملية الترميز وفك الترميز للبيانات، وتوليد واجهات RPC تلقائياً، مما يقلل من الجهد اليدوي في كتابة الكود ويحسن الكفاءة.

2. اشرح بإيجاز مفهوم "نمط البناء" المستخدم في مثال التعليمات البرمجية المقدم؟

نمط البناء (Builder Pattern) هو نمط تصميم يُستخدم في Protocol Buffers لإنشاء كائنات الرسائل (messages) بشكل آمن ومرن. في الأمثلة البرمجية الشائعة، مثل بناء رسالة Person، يتم استخدام فئة Builder (مثل Person.Builder) في Java أو Python لتعيين قيم الحقول تدريجياً (مثل setName() أو setAge()) قبل استدعاء build() لإنشاء الكائن النهائي. هذا النمط يتجنب مشكلات الـ mutable objects، يدعم السلاسل (chaining) للقراءة الجيدة، ويضمن أن الكائن يكون غير قابل للتعديل (immutable) بعد الإنشاء، مما يعزز السلامة في البرمجة الـ MultiThreading.

3. كيف تساعد Protocol Buffers في تقليل حجم البيانات المرسلة عبر السلك مقارنة بتمثيل JSON؟

Protocol Buffers (Protobuf) يقلل حجم البيانات من خلال استخدام تنسيق ثنائي (binary) بدلاً من النصي في JSON.

- ترميز فعال: يستخدم تقنيات مثل Varints للأرقام (ترميز متغير الطول يوفر مساحة للقيم الصغيرة)، ZigZag للأرقام السالبة، وطول محدد للسلاسل بدلاً من علامات اقتباس.

- عدم تضمين المفاتيح: في Protobuf، الحقول محددة برقم (field number) بدلاً من أسماء نصية كاملة كما في JSON، مما يقلل من التكرار.

- الضغط التلقائي: يدعم ضغط البيانات، وغالباً ما يكون الحجم أصغر بنسبة 3-10 مرات مقارنة بـ JSON، خاصة في البيانات الكبيرة أو المتكررة، مما يحسن سرعة النقل ويقلل من استهلاك الشبكة.

4. في مثال التعليمات البرمجية المقدم، ما أهمية تعيين قيم عددية (على سبيل المثال، name = 1، age = 2) للحقول داخل تعريف الرسالة؟

تعيين قيم عددية (field numbers) للحقول في تعريف الرسالة (مثل name = 1؛ age = 2) أمر أساسي في Protocol Buffers لأسباب عدة:

- الترميز الثنائي: تستخدم هذه الأرقام كمعرفات قصيرة في التنسيق الثنائي، مما يقلل من حجم البيانات مقارنة باستخدام أسماء نصية.

- التوافق الخلفي والأمامي: تسمح بإضافة أو إزالة حقول دون كسر التوافق؛ على سبيل المثال، يمكن لإصدارات قديمة تجاهل حقول جديدة بناءً على الأرقام غير المعروفة.

- الكفاءة في الفك: تساعد في تحليل البيانات بسرعة، حيث يقرأ المحلل الرقم أولاً لتحديد الحقل دون الحاجة إلى تخزين أسماء.

- القيود: يجب أن تكون الأرقام فريدة داخل الرسالة، ومن 1 إلى 536,870,911، مع حجز 19000-19999 للاستخدام الداخلي في Protobuf.

1. ما هو الغرض من الكلمة الأساسية repeated عند تعريف حقل في ملف proto؟

تستخدم لتعريف حقل يمكن أن يحتوي على قيم متعددة (مثل مصفوفة أو قائمة)، حيث يمكن أن يكون الحقل موجوداً صفر مره أو أكثر. على سبيل المثال، في تعريف رسالة، يمكن أن يكون repeated string names = 3؛ لتخزين قائمة من الأسماء. هذا يتيح تخزين عناصر متكررة بشكل فعال في التنسيق الثنائي، مع دعم للإضافة أو الإزالة دون تأثير على التوافق الخلفي، ويتم الوصول إليها عبر دوال مثل addNames() في الكود المولد.

2. ما ميزات بروتوكول HTTP2 وكيف وظيف مع grpc؟

ميزات بروتوكول HTTP/2 الرئيسية:

- التعددية (Multiplexing): يسمح بإرسال طلبات متعددة عبر اتصال واحد دون حظر، مما يحسن الكفاءة مقارنة بـ HTTP/1.1.

- الضغط (Header Compression): يستخدم HPACK لتقليل حجم الرؤوس المتكررة، مما يوفر عرض النطاق.

- الإطارات الثنائية (Binary Framing): يقسم البيانات إلى إطارات ثنائية للتحكم في التدفق والأولويات.

- دفع الخادم (Server Push): يسمح للخادم بإرسال موارد إضافية مسبقاً دون طلب.

- التحكم في التدفق (Flow Control): يدير تدفق البيانات لتجنب الازدحام.

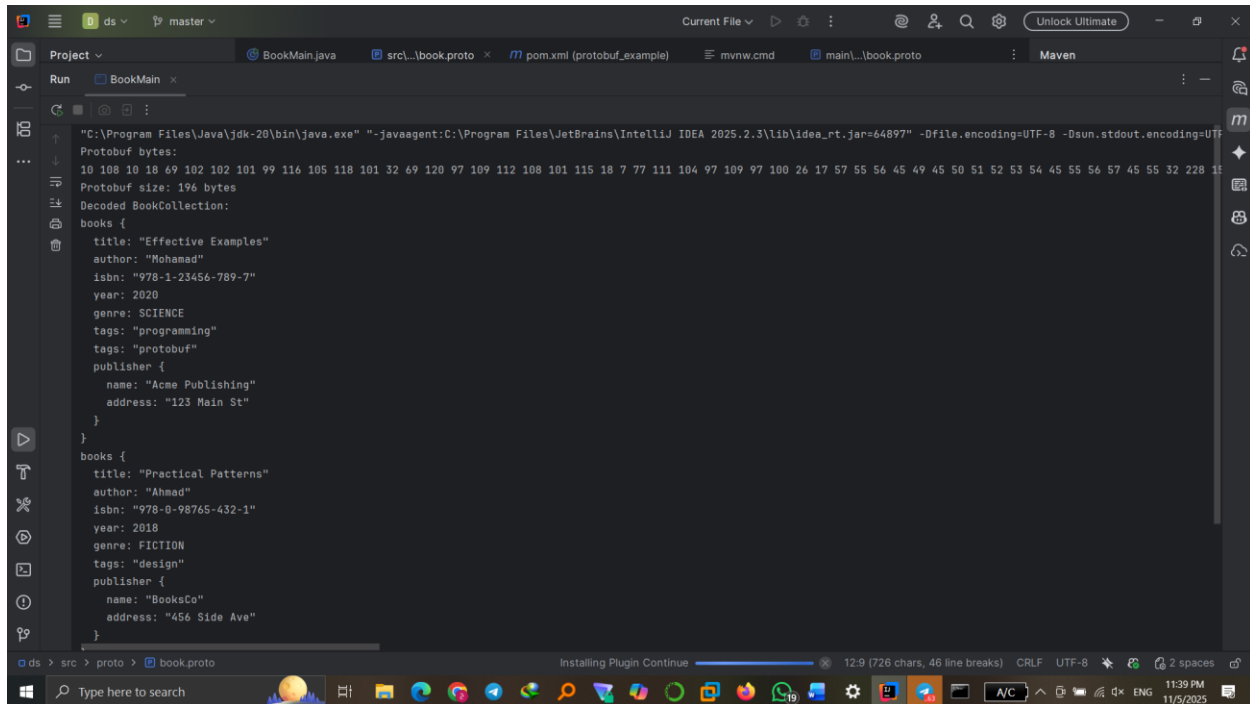
- الأمان: يتطلب عادةً TLS للتشفير.

كيف يُوظف مع gRPC؟

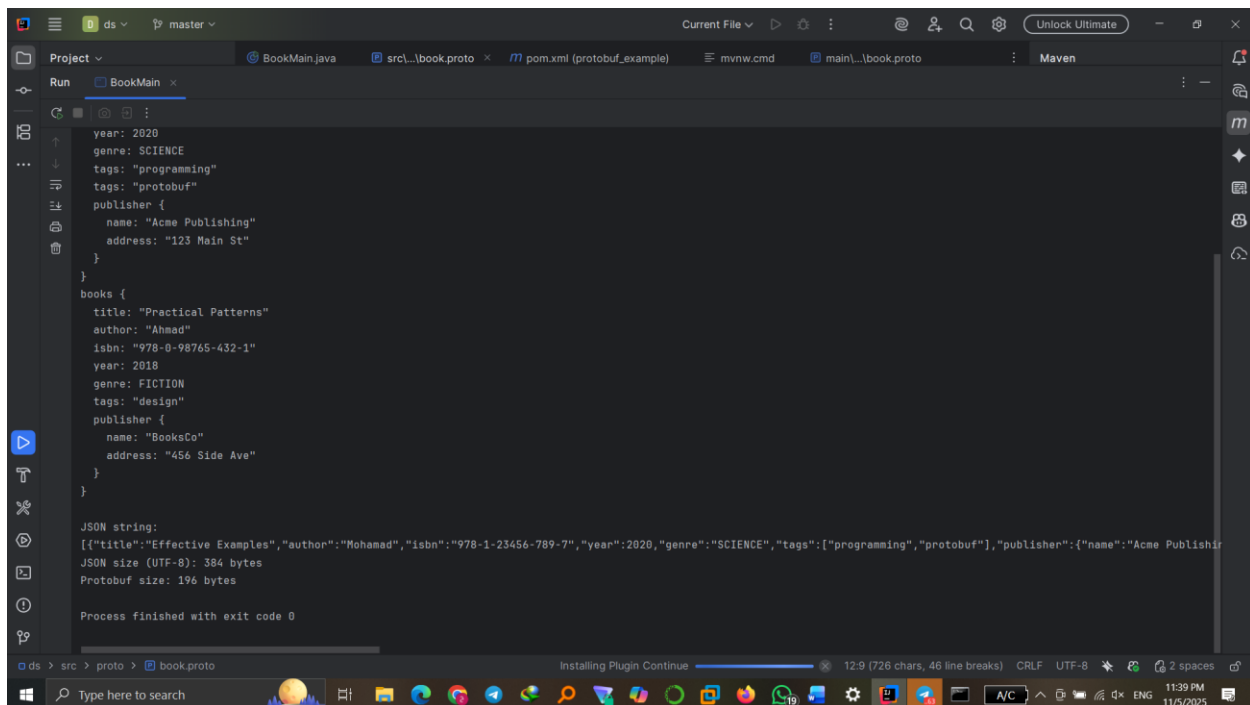
يُستخدم HTTP/2 كبروتوكول نقل أساسي في gRPC لدعم الاتصالات عالية الأداء. يستفيد gRPC من التعددية لإرسال تدفقات RPC متعددة (مثل unary، streaming) عبر اتصال واحد، مما يقلل من التأخير. كما يدعم الضغط والإطارات الثنائية لنقل بيانات Protobuf بكفاءة، بالإضافة إلى المصادقة والأمان عبر TLS. هذا يجعل gRPC مناسبًا للخدمات الداخلية والتطبيقات ذات الحمل العالي، حيث يحول الطلبات إلى رسائل HTTP/2 مع رؤوس محددة مثل grpc-status للنتائج.

Practical Implementation:

Example of executing for one time:



```
"C:\Program Files\Java\jdk-20\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.2.3\lib\idea_rt.jar=64897" -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8
Protobuf bytes:
10 108 10 18 69 102 102 101 99 116 105 118 101 32 69 120 97 109 112 108 101 115 18 7 77 111 104 97 109 97 100 26 17 57 55 56 45 49 45 50 51 52 53 54 45 55 56 57 45 55 32 228 15
Protobuf size: 196 bytes
Decoded BookCollection:
books {
  title: "Effective Examples"
  author: "Mohamad"
  isbn: "978-1-23456-789-7"
  year: 2020
  genre: SCIENCE
  tags: "programming"
  tags: "protobuf"
  publisher {
    name: "Acme Publishing"
    address: "123 Main St"
  }
}
books {
  title: "Practical Patterns"
  author: "Ahmad"
  isbn: "978-0-98765-432-1"
  year: 2018
  genre: FICTION
  tags: "design"
  publisher {
    name: "BooksCo"
    address: "456 Side Ave"
  }
}
```



```
year: 2020
genre: SCIENCE
tags: "programming"
tags: "protobuf"
publisher {
  name: "Acme Publishing"
  address: "123 Main St"
}
books {
  title: "Practical Patterns"
  author: "Ahmad"
  isbn: "978-0-98765-432-1"
  year: 2018
  genre: FICTION
  tags: "design"
  publisher {
    name: "BooksCo"
    address: "456 Side Ave"
  }
}
JSON string:
[{"title":"Effective Examples","author":"Mohamad","isbn":"978-1-23456-789-7","year":2020,"genre":"SCIENCE","tags":["programming","protobuf"],"publisher":{"name":"Acme Publishi
JSON size (UTF-8): 384 bytes
Protobuf size: 196 bytes
Process finished with exit code 0
```

For better readability:

```
"C:\Program Files\Java\jdk-20\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA
2025.2.3\lib\idea_rt.jar=64897" -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -
Dsun.stderr.encoding=UTF-8 -classpath "C:\Users\Hussein
Salloum\Desktop\5\ds\ds\target\classes;C:\Users\Hussein
Salloum\.m2\repository\com\google\protobuf\protobuf-java\4.28.2\protobuf-java-4.28.2.jar"
org.ds.proto.BookMain
```

Protobuf bytes:

```
10 108 10 18 69 102 102 101 99 116 105 118 101 32 69 120 97 109 112 108 101 115 18 7 77
111 104 97 109 97 100 26 17 57 55 56 45 49 45 50 51 52 53 54 45 55 56 57 45 55 32 228 15 40 3
50 11 112 114 111 103 114 97 109 109 105 110 103 50 8 112 114 111 116 111 98 117 102 58 30
10 15 65 99 109 101 32 80 117 98 108 105 115 104 105 110 103 18 11 49 50 51 32 77 97 105
110 32 83 116 10 84 10 18 80 114 97 99 116 105 99 97 108 32 80 97 116 116 101 114 110 115
18 5 65 104 109 97 100 26 17 57 55 56 45 48 45 57 56 55 54 53 45 52 51 50 45 49 32 226 15 40
1 50 6 100 101 115 105 103 110 58 23 10 7 66 111 111 107 115 67 111 18 12 52 53 54 32 83
105 100 101 32 65 118 101
```

Protobuf size: 196 bytes

Decoded BookCollection:

books {

title: "Effective Examples"

author: "Mohamad"

isbn: "978-1-23456-789-7"

year: 2020

genre: SCIENCE

tags: "programming"

tags: "protobuf"

publisher {

name: "Acme Publishing"

address: "123 Main St"

}

}

```
books {  
  title: "Practical Patterns"  
  author: "Ahmad"  
  isbn: "978-0-98765-432-1"  
  year: 2018  
  genre: FICTION  
  tags: "design"  
  publisher {  
    name: "BooksCo"  
    address: "456 Side Ave"  
  }  
}
```

JSON string:

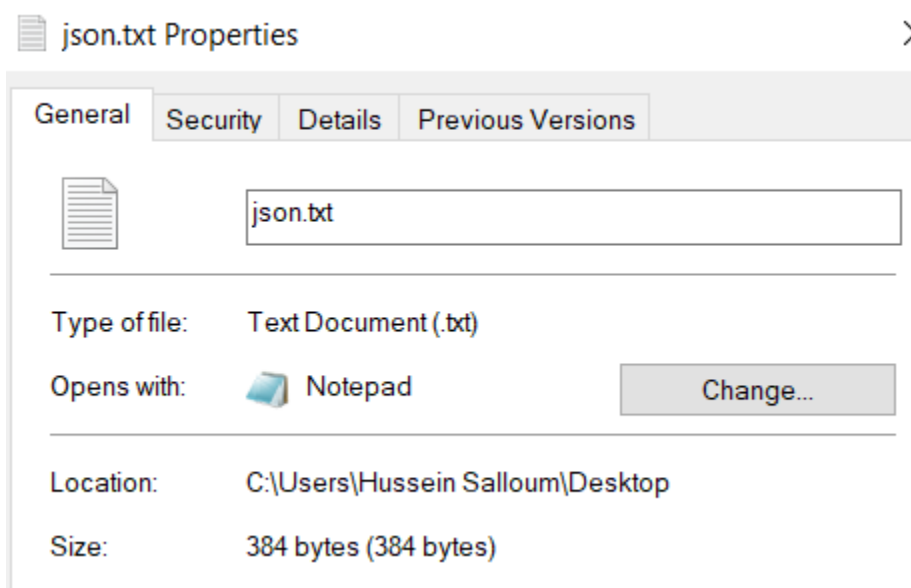
```
[{"title":"Effective Examples","author":"Mohamad","isbn":"978-1-23456-789-7","year":2020,"genre":"SCIENCE","tags":["programming","protobuf"],"publisher":{"name":"Acme Publishing","address":"123 Main St"}}, {"title":"Practical Patterns","author":"Ahmad","isbn":"978-0-98765-432-1","year":2018,"genre":"FICTION","tags":["design"],"publisher":{"name":"BooksCo","address":"456 Side Ave"}}]
```

JSON size (UTF-8): 384 bytes

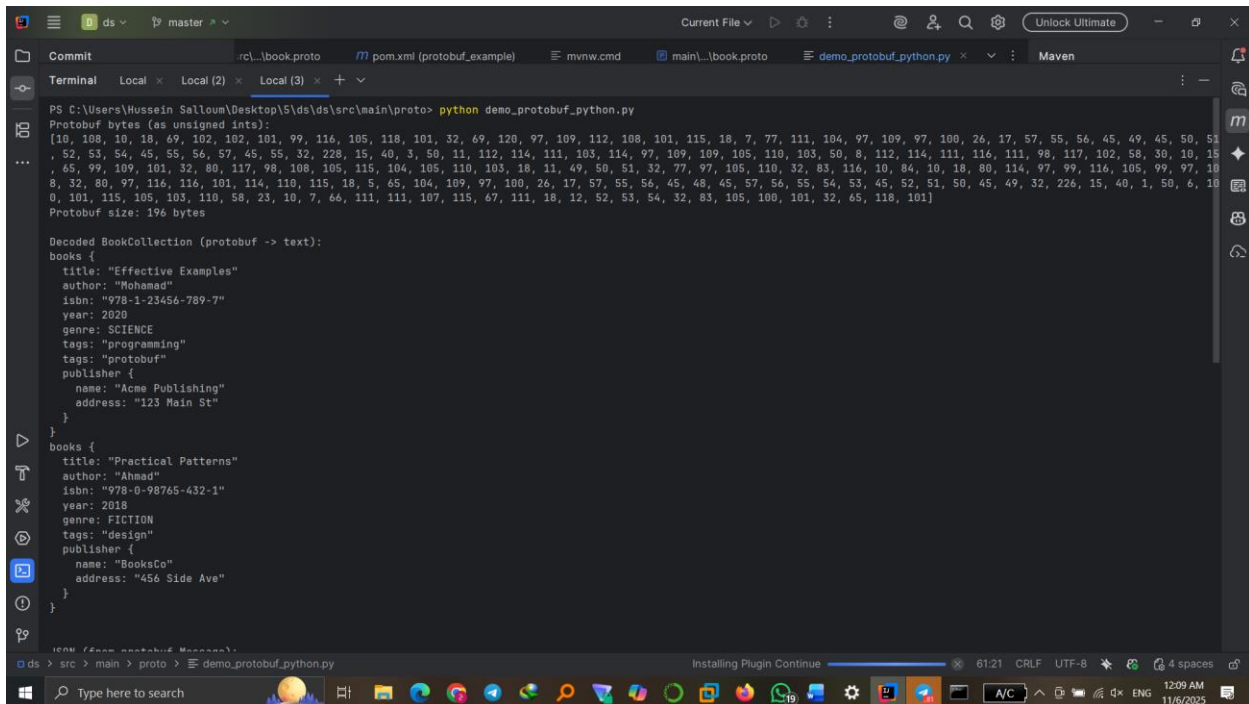
Protobuf size: 196 bytes

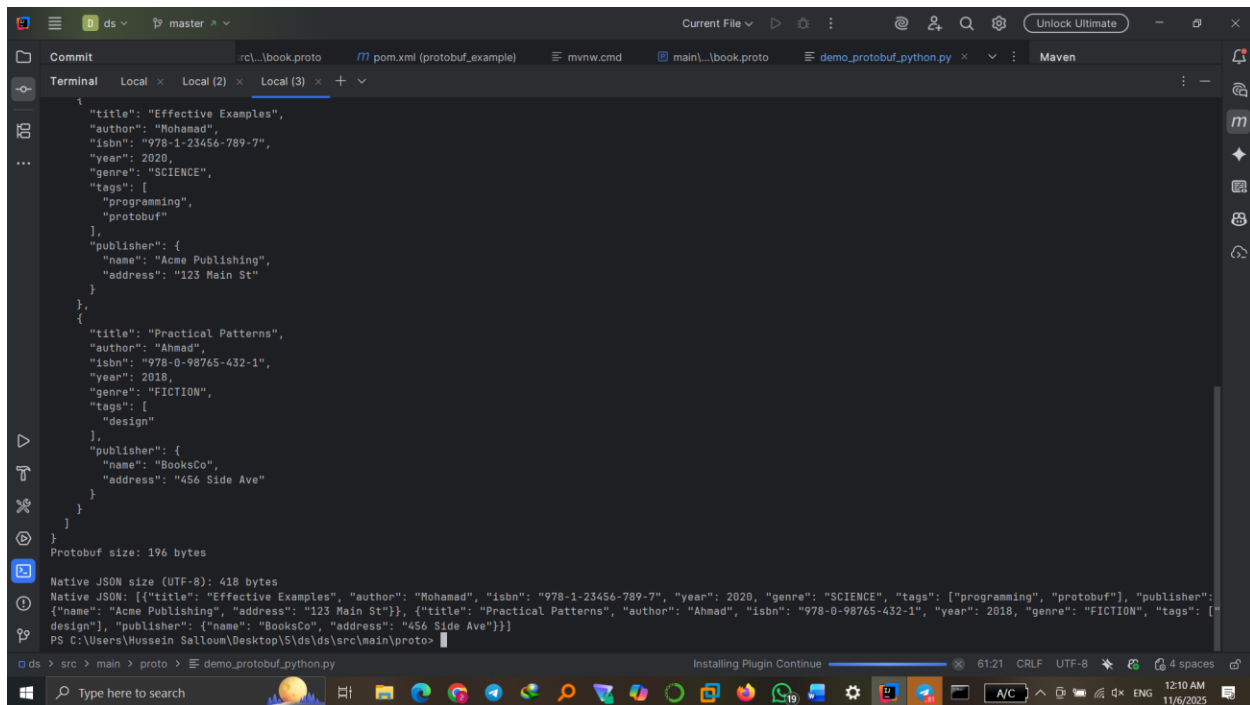
Process finished with exit code 0

To make sure about the json size we put the json in a text file and see its size:



4- for python:





For better readability:

```
PS C:\Users\Hussein Salloum\Desktop\5\ds\ds\src\main\proto> python
demo_protobuf_python.py
```

Protobuf bytes (as unsigned ints):

[10, 108, 10, 18, 69, 102, 102, 101, 99, 116, 105, 118, 101, 32, 69, 120, 97, 109, 112, 108, 101, 115, 18, 7, 77, 111, 104, 97, 109, 97, 100, 26, 17, 57, 55, 56, 45, 49, 45, 50, 51, 52, 53, 54, 45, 55, 56, 57, 45, 55, 32, 228, 15, 40, 3, 50, 11, 112, 114, 111, 103, 114, 97, 109, 109, 105, 110, 103, 50, 8, 112, 114, 111, 116, 111, 98, 117, 102, 58, 30, 10, 15, 65, 99, 109, 101, 32, 80, 117, 98, 108, 105, 115, 104, 105, 110, 103, 18, 11, 49, 50, 51, 32, 77, 97, 105, 110, 32, 83, 116, 10, 84, 10, 18, 80, 114, 97, 99, 116, 105, 99, 97, 108, 32, 80, 97, 116, 116, 101, 114, 110, 115, 18, 5, 65, 104, 109, 97, 100, 26, 17, 57, 55, 56, 45, 48, 45, 57, 56, 55, 54, 53, 45, 52, 51, 50, 45, 49, 32, 226, 15, 40, 1, 50, 6, 100, 101, 115, 105, 103, 110, 58, 23, 10, 7, 66, 111, 111, 107, 115, 67, 111, 18, 12, 52, 53, 54, 32, 83, 105, 100, 101, 32, 65, 118, 101]

Protobuf size: 196 bytes

Decoded BookCollection (protobuf -> text):

books {

title: "Effective Examples"

```
author: "Mohamad"
isbn: "978-1-23456-789-7"
year: 2020
genre: SCIENCE
tags: "programming"
tags: "protobuf"
publisher {
  name: "Acme Publishing"
  address: "123 Main St"
}
}
books {
  title: "Practical Patterns"
  author: "Ahmad"
  isbn: "978-0-98765-432-1"
  year: 2018
  genre: FICTION
  tags: "design"
  publisher {
    name: "BooksCo"
    address: "456 Side Ave"
  }
}
```

JSON (from protobuf Message):

```

{
  "books": [
    {
      "title": "Effective Examples",
      "author": "Mohamad",
      "isbn": "978-1-23456-789-7",
      "year": 2020,
      "genre": "SCIENCE",
      "tags": [
        "programming",
        "protobuf"
      ],
      "publisher": {
        "name": "Acme Publishing",
        "address": "123 Main St"
      }
    },
    {
      "title": "Practical Patterns",
      "author": "Ahmad",
      "isbn": "978-0-98765-432-1",
      "year": 2018,
      "genre": "FICTION",
      "tags": [
        "design"
      ],
    },
  ]
}

```

```

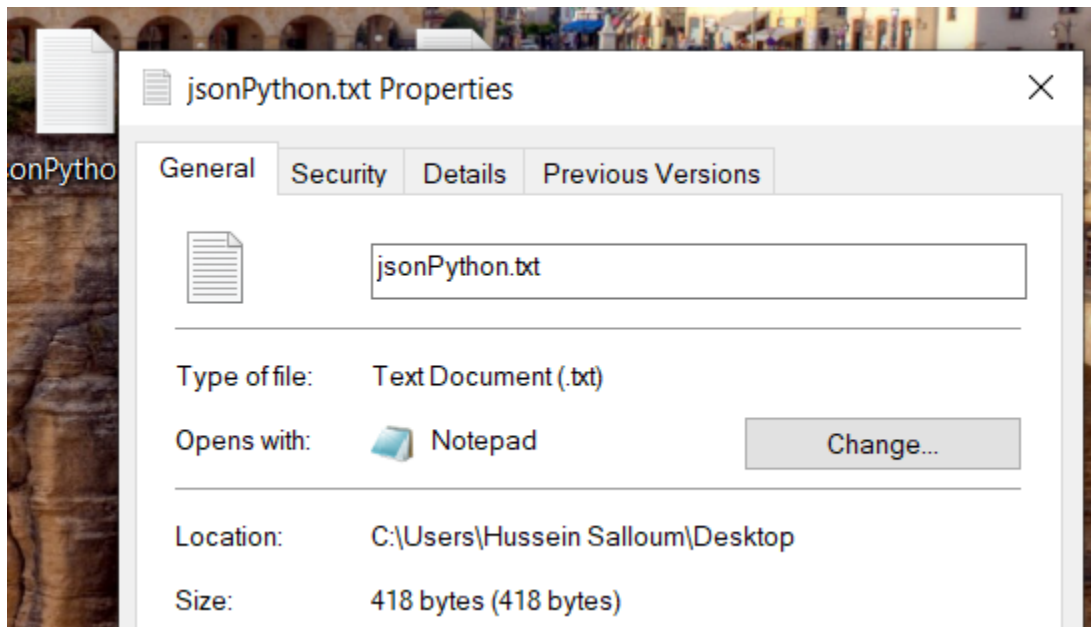
"publisher": {
  "name": "BooksCo",
  "address": "456 Side Ave"
}
}
]
}

```

Protobuf size: 196 bytes

Native JSON size (UTF-8): 418 bytes

Native JSON: [{"title": "Effective Examples", "author": "Mohamad", "isbn": "978-1-23456-789-7", "year": 2020, "genre": "SCIENCE", "tags": ["programming", "protobuf"], "publisher": {"name": "Acme Publishing", "address": "123 Main St"}}, {"title": "Practical Patterns", "author": "Ahmad", "isbn": "978-0-98765-432-1", "year": 2018, "genre": "FICTION", "tags": ["design"], "publisher": {"name": "BooksCo", "address": "456 Side Ave"}}]



See the size is 418 byte as calculated

We notice a noticeable difference in the size of data transferred between json and protobuf bytes so protobuf is better than restful in sending data.

We notice that protobuf worked for java and python with the same benefits so it is independent from the programming languages

Actually the json text is same in the examples for java and python but in python code it just add some spaces which increased some bytes but ofcourse they are same 384 bytes if we deleted all the spaces.

Ofcourse we could fix the python code to don't do that but the idea here is protobuf is independent from the PL that's all.

Additional exercise:

In python we created server and client

The server starts on port 50051 waits for requests

```
PS C:\Users\Hussein Salloum\Desktop\5\ds\ds\src\main\proto> python server.py
gRPC Server Running on port 50051...
```

Client:

```
PS C:\Users\Hussein Salloum\Desktop\5\ds\ds\src\main\proto> python client.py
CreateBook Success: True
Book Received:
title: "Clean Code"
author: "Robert Martin"
isbn: "9780132350884"
year: 2008
PS C:\Users\Hussein Salloum\Desktop\5\ds\ds\src\main\proto>
```

Client connected to server through grpc, send createbook then the server stored this book so when the client requested getbook the client got it from the server

The data were sent in protobuf so in binary so speedier and lighter.

