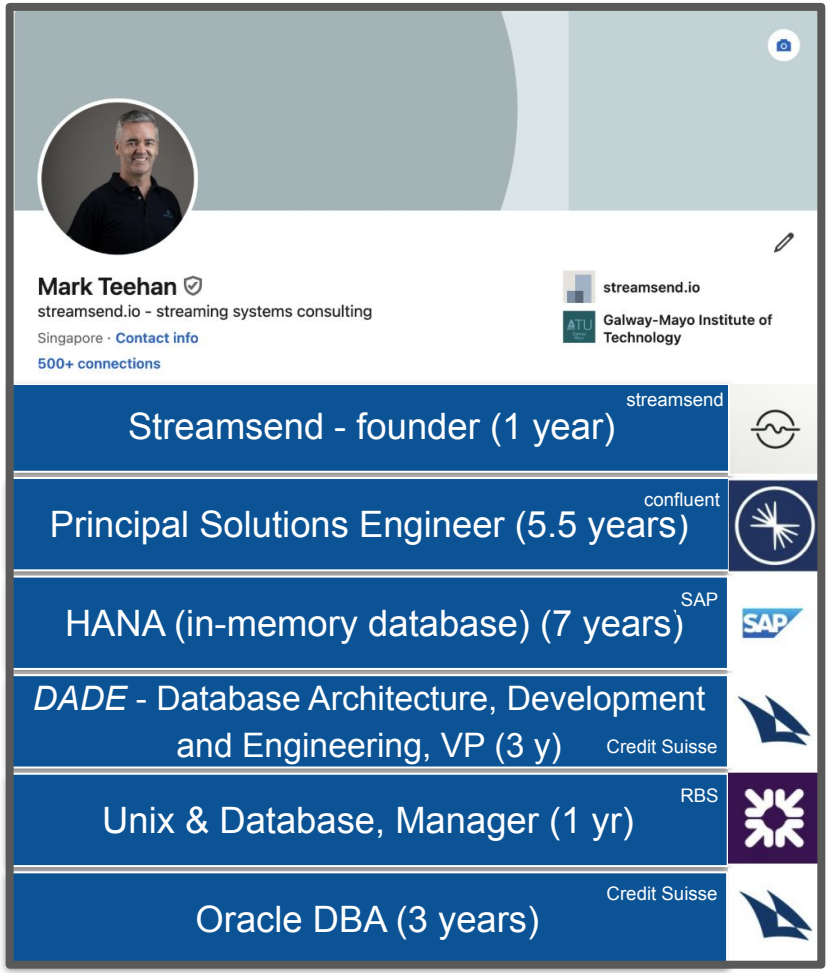
Me

I do *Data Streaming* Systems

- *I don't speak any Mandarin*
- *I worked for Confluent until October 2023*
- *Streamsend file-streaming is free*

Now I live in Taipei
Taiwan Gold Card

- *Songshan district*
- *Office is a Louisa*
- *Coding using Rust & librdkafka*



Use Kafka for File Streaming

1. Introduction
2. Project Origin
3. The Problem to Solve
4. Release 1 and Release 2
5. Using Kafka to Stream Files alongside events
6. How Uploading works
7. How Downloading works
8. Quick Demo
9. File streaming - when to [not] use it
10. Some more Kafka stuff

PBS NEWS

Menu

Full Episodes | Podcasts | Newsletters | Live

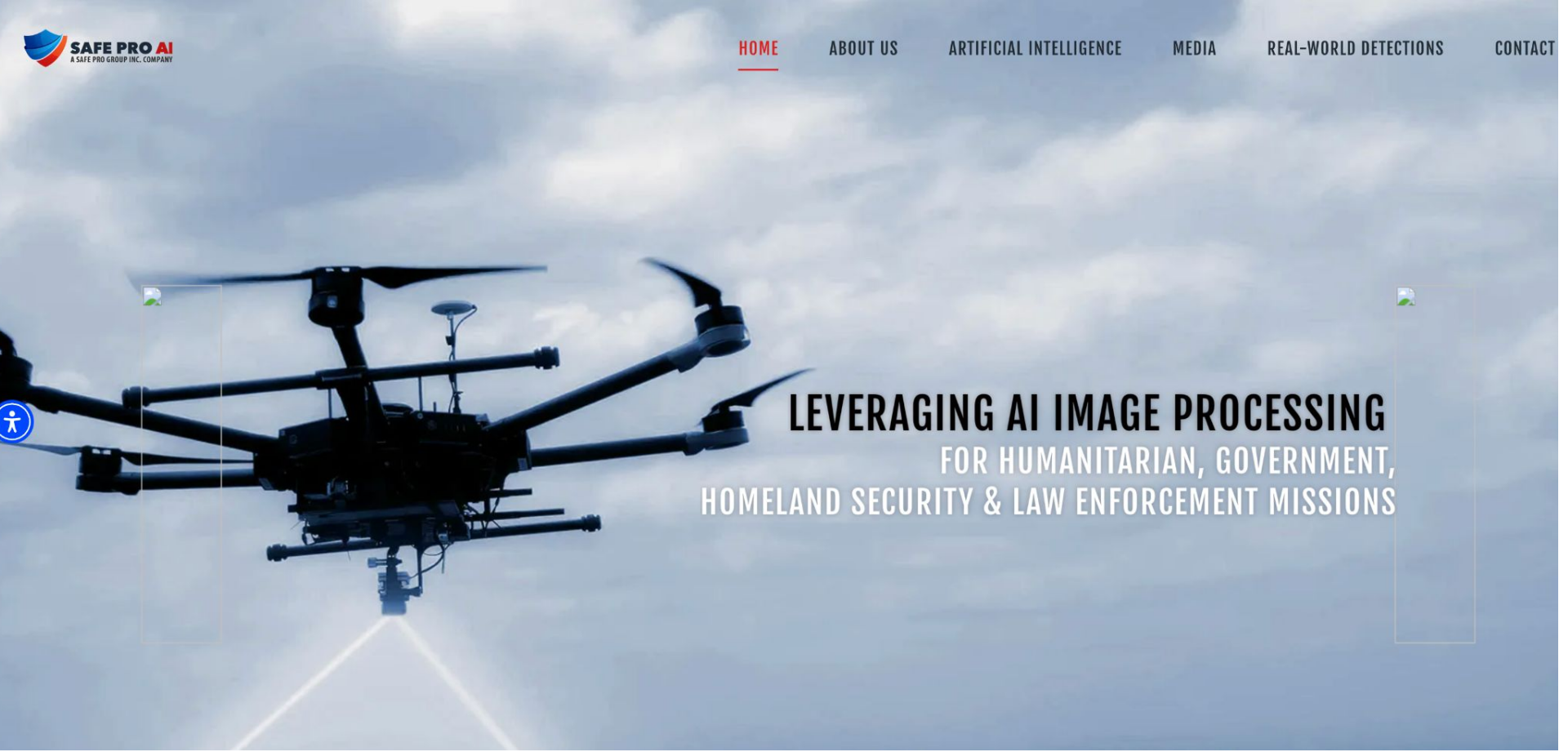
PBS NEWS HOUR

New drone technology could make it easier to clear unexploded bombs, mines in Ukraine

Sep 14, 2023 6:35 PM EDT

Related

Support P



https://safeproai.com

SAFE PRO AI

SAMPLE OF REAL-WORLD LANDMINE & UXO DETECTIONS | MAY 2025

Object Detected: TM-62 Anti-Tank Mine

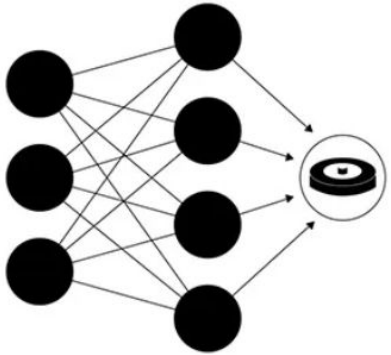
Object Detected: Artillery Shell

Object Detected: 40mm VOG-25 Grenade

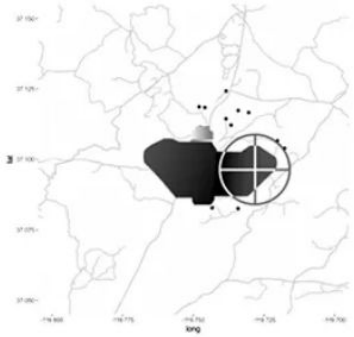
SAFE PRO AI

UAV Survey

Machine Learning



Mapping and Detection



Cost Effective
Prioritize areas of clearance - focus resources on mined areas

Safer
Improve situational awareness & detect seismically activated mines

Faster
Scan acres of land in minutes

Effective
Current algorithm detects 80-90% of surface mines and munitions

SAFE PRO AI

The Problem

- Drones capture thousands of images
- Each image is ~30MB
- Images must be uploaded to the Safe Pro AI platform for analysis
- Various challenges:
  - Network Quality
  - Time Pressure
  - Operating environment Safety
  - Remote locations

2023: Research a **Kafka Data Pipeline** to Stream Images

The Research

- Kafka **max.message.bytes** is 1MB
- 2018-2023: Mark (Confluent Sales Guy): “Oh it’s easy, just *Chunk the File* ….”
- How to do this?
- Develop Kafka Connect plugins `file-chunk-source` & `file-chunk-sink`
- Evaluated and Tested, Improved, Re-released
- Listed on Confluent Hub
- Safe Pro AI **adopted a different data pipeline technology** that addresses various other challenges for the field-team.
- file-streaming *is a fancy way to send a file*. I think it is interesting…

Release 1: Kafka Connect file-chunk-source and file-chunk-sink

Release 1 Kafka Connect file-chunk-source and file-chunk-sink

- Source Connector:
  - Monitor a filesystem for new files
  - Split files into chunks (`file.chunk.size.bytes`)
  - Produce chunks to a topic, ordered
- Sink Connector
  - Consume from the topic
  - Append chunks to a file
  - Run a MD5 checksum

Why Kafka?

- **Infinite Retries**
- **Encryption Ciphers**
- **Compression**
- **Open Source**
- **SaaS or Self-managed**

Why not Kafka?

- **AK Client Size unsuitable for edge devices**
- **JVM memory requirements limit file size**
- **Filesystems prevent clustering - single-task mode only**

2025: Release 2 Rewrite using Rust

- Source Connector is now the **Uploader**
  - Rust is faster: no GC, better threading, useful rust crates
  - <20MB executable - ok for small devices (vehicles, point of sale, manufacturing)
  - In memory: 100% stateless
  - Handles larger files: (uses swap, not JVM): streamed a 27GB file on a 16GB macbook
  - Uses librdkafka to produce/consume
  - Detects Cluster/Topic max.message.bytes and automatically sets the chunk.size.bytes
  - Run one or many Uploaders to one (or many) Downloaders
- Sink Connector is now the **Downloader**
  - Consume and merge and verify with MD5: all in-memory
  - Handles hundreds of Uploaders at once
  - Mirror files by starting additional Downloaders

It’s a pet-project….

- **Free** Edition
  - Unlimited deployment of uploaders and downloaders
  - Produces to the first partition of the topic
  - Fast enough for almost every use case
  - Works on Apache Kafka or any Confluent - SASL/SSL or noauth or AWS IAM
  - Linux-AMD64 command line, Linux-AMD64 Docker, Macos cmd-line
  - Deploy on kubernetes
- **Pay** Edition
  - Produces to all partitions of the topic
  - Very high throughput capacity
  - Read/Write to S3 (soon)
  - Scheduled Streaming (soon)

streamsend/downloader

By streamsend • Updated 3 days ago

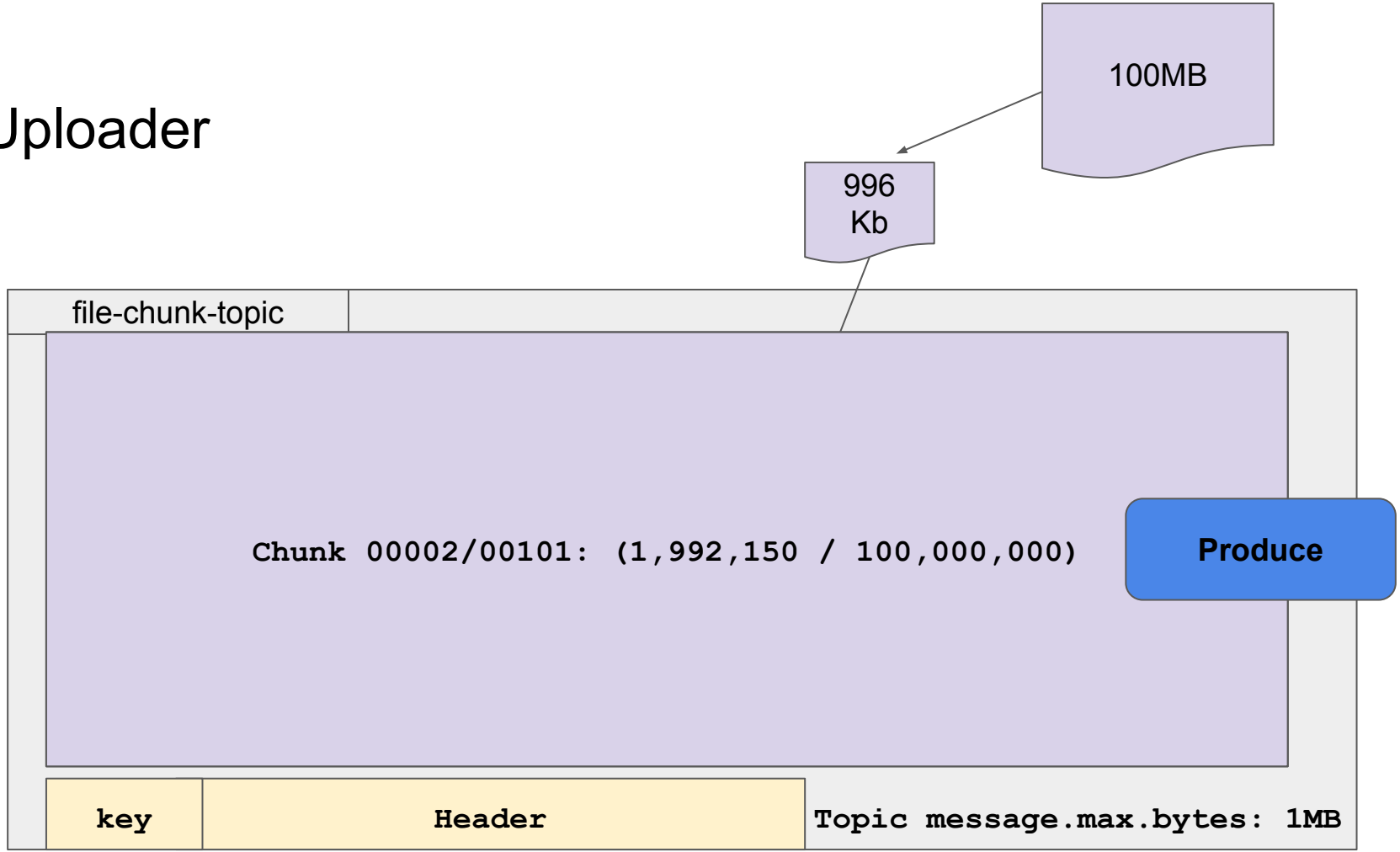
Accompanies streamsend/uploader to recreate files from the event-sized chunks

streamsend/uploader

By streamsend • Updated 3 days ago

Stream any file through a Kafka topic by splitting a file into event-sized chunks.

Uploader


100MB
996
Kb
file-chunk-topic
Chunk 00002/00101: (1,992,150 / 100,000,000)
Produce
key
Header
Topic message.max.bytes: 1MB

Uploader

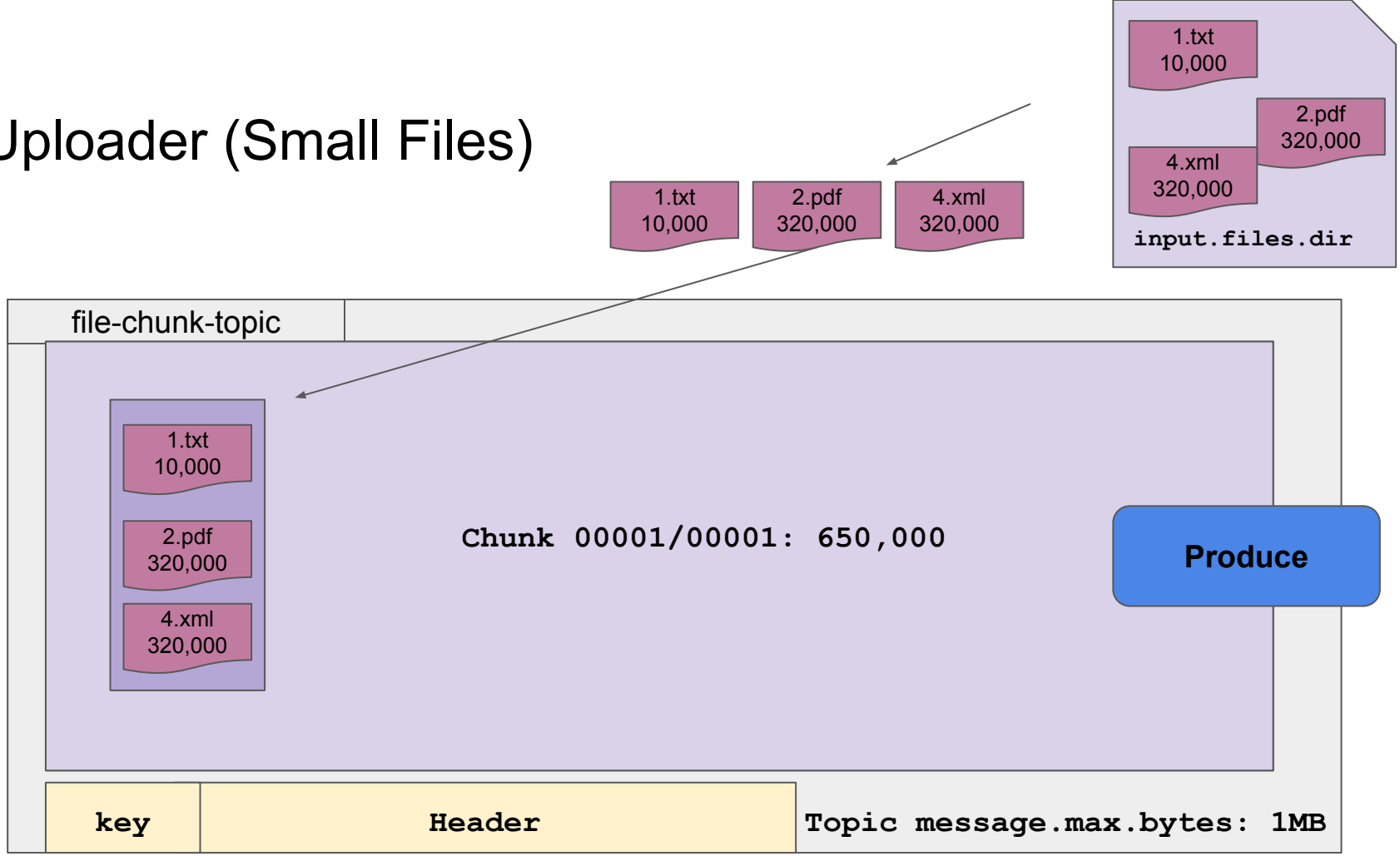
Uploader continues to produce chunks of the file to the topic partition
The last chunk is ~smaller than the other chunks

```
 batch.size = 1, linger.ms=5ms, retries=infinite,
compression=none
```

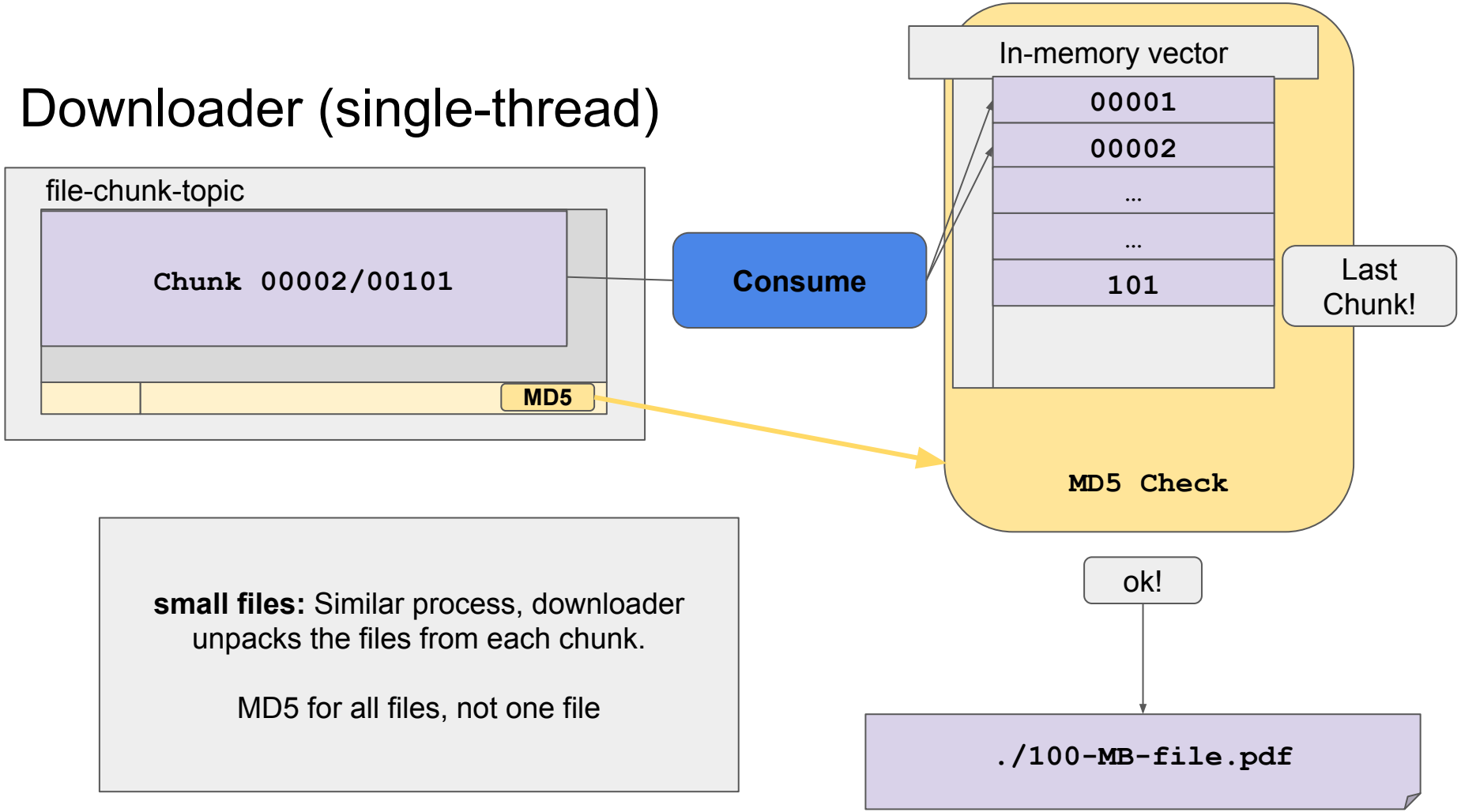
The **Message header** contains:

- File name, size, creation-time
- Subdirectory (..subdirectories are recreated by Downloader)
- MD5 checksum
- Chunk ID, Number of Chunks, [chunksets]

Uploader (Small Files)



Downloader (single-thread)



small files: Similar process, downloader unpacks the files from each chunk.

MD5 for all files, not one file

https://streamsend.io/#/

…why *Stream* a File

I can use cp
(or scp or rsync or ftp
....or torrent)

Context matters.
There should be a good reason

Some Good Reasons:

Files Alongside Events
An event pipeline already exists
“But there are some files …”
Use: if you already have Kafka in Prod

Streaming Protocol
Secure & efficient: retries, in.flight,
Compression, Encryption
Use for : Network / Edge / Capacity

Context matters.
There should be a good reason

Scalability
Partitions, Consumer Groups
Bittorrent for the Enterprise?
Use: Large volume; manufacturing, retail

…why **not** *Stream* a File

If cp or scp or rsync or ftp
works ok,
keep it

I am just talking about static files

Use streaming media protocols for streaming media
(Youtube, netflix etc).

Closed files; not appending/changing files

Not for Database Backups

No Processing: you cannot look inside the file

(....for now)

Why File Streaming in interesting

File senders (cp, scp etc) are serialized, single-threaded, first-to-last byte

Kafka is a common enterprise platform, and most clusters operate below capacity.

The Kafka protocol is (arguably) the best part of Kafka

It is fun and interesting to explore what-else it could do

The Kafka ecosystems is experiencing lots of “break-out” projects (S3, messaging, Agentic)

File streaming is another break-out project (I think)

It could be the-fastest-way to send a file; point to point.

Some Kafka Stuff

Coded in rust, lib is librdkafka

Consumer offsets are not used because files are immutable - replay is simpler than managing offsets

Uses direct partition assignment rather than keys to ensure ordering

Message value encoding is bytestream: no plans for serialized BLOBs: no point….

Multi-threaded option uses all partitions, a consumer group and an ordering algorithm

Topic 1: chunks - retention should match expected recovery period

Topic 2: state (compacted) - each Uploader records its state in one message

Uploaders monitor for files between file.minimum.age.ms and file.maximum.age.ms

Take-aways

Files-alongside-events, using the same pipeline, is possible

Streamsend is free - I will provide support, time-permitting. Please try it out!

This is the only file-streaming product for Kafka

Retail: hundreds of point-of-sale uploaders (images, documents, video) to one downloader

Manufacturing: send images from manufacturing machines into HDFS or S3

Drones/Vehicles/Devices: streaming using small-footprint, secure, infinity-retry pipelines

Next..

- S3 Uploaders & S3 Downloaders (stream from many devices to/from S3)
- Flink: Stream-Process instead of Download
- Uploader releases for Windows & ARM64
- High performance: Uploader scales to the capacity of the network interface with n partition thread
- Add storage-types for Azure Cloud Storage and Google Object Storage
- Scheduled Uploaders: use cron strings (“00,30 * * * *”) to schedule Uploader streams
- …

Me

謝謝！

Please add me on LinkedIn (Mark Teehan)
Or
Email: mark.teehan@streamsend.io
Or
Meet at Louisa Qingchang St, Songshan