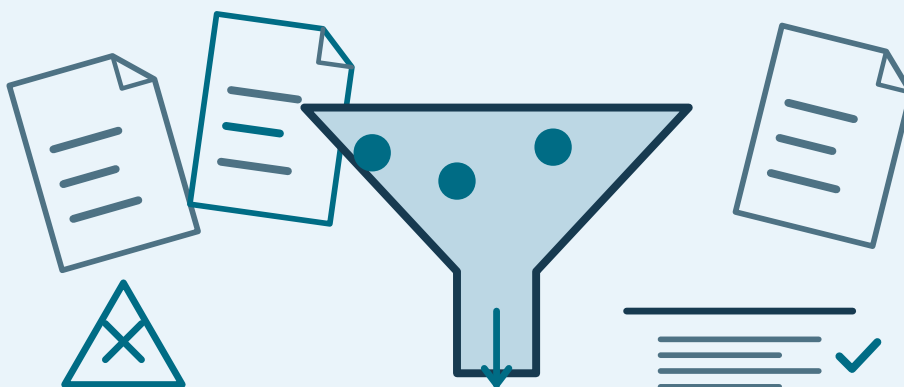




PYTHON AND DATA WORKFLOWS

Build Reliable File Ingestion Pipelines

Use manifests, content fingerprints and atomic file acceptance to make retries explainable



Sanjaay Gowda T S

FREE SAMPLE / INTRODUCTION + CHAPTER 1 / EDITION 1.0.0

sanjaaygowdats.ai

About this free sample

Build Reliable File Ingestion Pipelines

Sanjaay Gowda T S

First edition 1.0.0, 2026-10-13

Copyright 2026 Sanjaay Gowda T S. This is a labelled free sample, not the complete ebook. You may share this sample unchanged. It contains the cover, sample permissions, contents, introduction, Chapter 1 and a closing product link. The remaining chapters, worked solutions and companion files are not included. The full ebook text, diagrams and artwork have separate single-reader terms; this sample permission does not authorize sharing the full purchased PDF or EPUB.

The companion source code, tests and fictional data are separately MIT licensed. You may adapt or share that project with its included licence notice. MIT does not apply to the ebook text or book artwork. The books and companion projects have no technical activation requirement or buyer-specific watermark.

This first edition was prepared with AI assistance. The case studies are fictional teaching examples. Any measured outputs come from the supplied examples, not production customer data. The teaching scope, assumptions and limitations are explained in the chapters. Independent human technical, editorial and assistive-technology reader review is pending. Automated tests and accessibility checks do not establish certification or suitability for every use.

Run examples in a separate working folder using the supplied synthetic data. Review inputs and calculations before adapting the techniques to real operations. Model software, service access, subscriptions, ongoing updates and consulting are not included.

Author website: sanjaaygowdats.ai

Ebook terms: sanjaaygowdats.ai/ebooks/license

Inside this free sample

Start here.....	4
1. Define the ingestion promise.....	5
Continue with the full guide	6

This free sample contains the introduction and Chapter 1 only. The closing link opens the complete guide. The companion project and worked solutions are supplied with the complete ebook.

Start here

A file import becomes a pipeline when it has to survive repeated deliveries, malformed records and interruptions without making its history unintelligible. The central question is not how to read a directory. It is when you are allowed to say that a delivery has been accepted, what that statement means and how a later retry will be interpreted.

This book uses Alder Depot, a fictional warehouse that receives small arrival-event files. Each JSON Lines record contains an event identity and a nonnegative unit count. All names, paths, records and quantities are invented. The program does not model inventory accounting or claim that a recorded arrival proves physical goods were received. It teaches a narrow ingestion boundary with explicit evidence and failure states.

The companion is a deterministic ingestion planner. Its JSON input contains file paths and file contents as strings, plus a previous receipt ledger and previously accepted event identities. It returns accepted events and the proposed next ledger. It deliberately does not read arbitrary disk paths or persist state. This makes retry behavior easy to inspect while keeping filesystem permissions and crash recovery outside the runnable lesson's claims.

```
python main.py --input example-input.json
python -m unittest discover -s tests -v
```

The fixture has three deliveries. The first and third are valid. The second contains one valid-looking event followed by invalid JSON. None of that second file's events should be accepted. Run the tests, then feed the proposed ledger and event identities into a second run to observe an identical retry. Keep the original fixture unchanged and make experiments in a separate copy.

The book separates three identities, explains atomic file acceptance and designs the durable boundary that a real adapter would need. It does not claim exactly-once delivery across distributed systems. You will learn to state a narrower promise that you can actually test, then identify the additional persistence and concurrency work required to extend it.

1. Define the ingestion promise

Acceptance is a state transition

Alder accepts a file only when its path is permitted, its content is new for that path, every record matches the event contract and every event identity is new. Acceptance adds all events and records the content fingerprint together in the returned plan. Rejection adds neither. An identical retry of a previously accepted path and fingerprint adds no new events.

The file is the atomic validation unit. This choice is stricter than accepting good rows from a mixed file. It makes a retry easy to explain because the producer can correct and resubmit the whole rejected delivery. A row-level policy is also possible, but it needs stable record-level receipts so a retry does not duplicate the rows accepted earlier.

State	Meaning	Proposed event effect
Accepted	Entire new file passed	All file events added
Skipped	Same path and fingerprint	No new events
Rejected	Contract or conflict failed	No file events added

These states refer to the planner's supplied evidence. They do not establish that an operational database commit succeeded. Calling the output `next_known` emphasizes that it is proposed state, not a durable receipt. The caller must not acknowledge a source delivery merely because the planner returned JSON.

Worked case: the half-valid file

The second fixture file starts with B101 and then contains `not-json`. A naive loop might insert B101 before reaching the error. If the source retries the corrected file, B101 could appear twice. Staging the entire file first avoids this partial acceptance. The output must contain neither B101 nor a receipt for that rejected path. The failure points to the second line so the producer has a concrete repair target.

Exercise: name the promise

Write one sentence describing the runnable program's guarantee and another describing what it does not persist. Include the unit of atomicity and the evidence used to recognize a retry. Avoid the phrase `exactly once` unless you specify the boundary, state store and concurrency assumptions that would make that phrase meaningful.

Acceptance worksheet: write the state explicitly

Before the first fixture run, the ledger and event history are empty. Afterward, the ledger contains the first and third paths, while event history contains A101, A102 and C101. B101 is absent despite appearing on a syntactically valid first line. Write these collections on paper before examining the program's output. Then ask whether each collection represents newly accepted work or all known work. The events output contains only newly accepted records; `next_known_event_ids` contains prior and new identities together. Confusing those two populations can cause a caller to reinsert old events on every run. A persistence adapter should make that distinction explicit in its own data model and command interface. If the destination already stores event bodies, it may derive the known identity set from the same durable table rather than maintaining an independently overwritten list. That reduces one opportunity for saved state to diverge.

Continue with the full guide

This free sample ends after the introduction and Chapter 1. The complete guide contains nine teaching chapters, worked exercise solutions, primary references and a separate Python companion project. The full PDF, reflowable EPUB and project files are supplied together.

Use the complete guide to work through the remaining exercises and inspect the supplied examples. The companion code and fictional data are separately MIT licensed; the complete ebook has the single-reader terms described on its product page.

The examples are teaching fixtures, not production customer results. Independent human technical, editorial and assistive-technology reader review remains pending. Software access, subscriptions, consulting and guaranteed results are not included.

[View the complete guide and purchase options.](#)

You may share this labelled free sample unchanged. Keep the full purchased PDF and EPUB private. Thank you for reading.