Retry Is Not Resume

A 30-Minute Interruption-Proof Web-to-CSV Lab

RUN → STOP → RESUME

Build the part most automation tutorials skip: durable progress that survives a terminated process.

10 / 10
EXTRACTED PRODUCT TESTS

0
COMPLETED RECORDS REFETCHED

30 min
OFFLINE LAB

The narrow promise

This is not a broad web-scraping book. It is a field manual for one expensive failure: a collection job stops after saving useful work, then starts from the beginning.

> **OUTCOME**
>
> By the end, you can explain, run, and inspect a pipeline that resumes after interruption without fetching completed records again.

You will build a mental model for

- ✓ the boundary between retry and resume;
- ✓ four files with four separate responsibilities;
- ✓ checkpoint ordering and its crash windows;
- ✓ a forced-stop test that proves the behavior;
- ✓ configuration drift and safe reset decisions.

The lab uses local HTML fixtures. It needs no paid API and makes no external requests.

One result, ten moves

A retry repairs an attempt. A resume repairs a run.

RETRY

Same record, same run

A request times out. The process is alive. Try that request again, with a limit and backoff.

RESUME

Next record, next run

The process is gone. A new process must discover durable progress and continue after it.

These are different failure boundaries. A ten-attempt retry loop cannot remember which records a dead process completed.

```
record B request fails
→ retry B inside the same process

process stops after saving record B
→ new process resumes at record C
```

THE DANGEROUS DEMO

A script that completes once proves only the happy path. It says nothing about what happens between two durable writes.

Name the boundary before choosing the fix

Transient request

Timeout, temporary server response, connection reset.

Tool: bounded retry

Process interruption

KeyboardInterrupt, crash, reboot, lost terminal.

Tool: durable checkpoint

Partial persistence

CSV changed but state did not, or a file was truncated.

Tool: atomic replace + ordering

Configuration drift

Fields, ID rules, or source contract changed.

Tool: configuration fingerprint

“Add retries” is correct for only one quadrant. Treating every failure as a request problem creates false confidence.

A useful incident question

What knowledge disappeared when the process died, and which file should have preserved it?

Do not make one file carry four meanings

`normalized.csv`

Deliverable truth

The usable rows a downstream consumer receives.

`state.json`

Resume truth

Completed IDs plus the configuration fingerprint.

`run_log.jsonl`

Event truth

Append-only facts for each attempt, skip, and failure.

`report.html`

Human summary

A readable view derived from the run evidence.

DESIGN RULE

If deleting the report breaks resume, the report owns too much. If reading the CSV is the only way to know progress, the output contract owns too much.

Separate responsibilities make recovery decisions inspectable. They also let you regenerate a report without changing progress.