

PaperCheck: how a one-person project became a three-cloud system

A tool that checks research papers and student reports for AI-sounding writing and for plagiarism. Built by one student in about a week, run on free tiers and student credit, and used by hundreds of students. This is the full architecture, how the pieces fit, what broke, and why the design looks the way it does.

311

accounts in the first
week

~190

checks a day, 9,000
words each on
average

~1M

words checked on the
busiest day

1 to 3 min

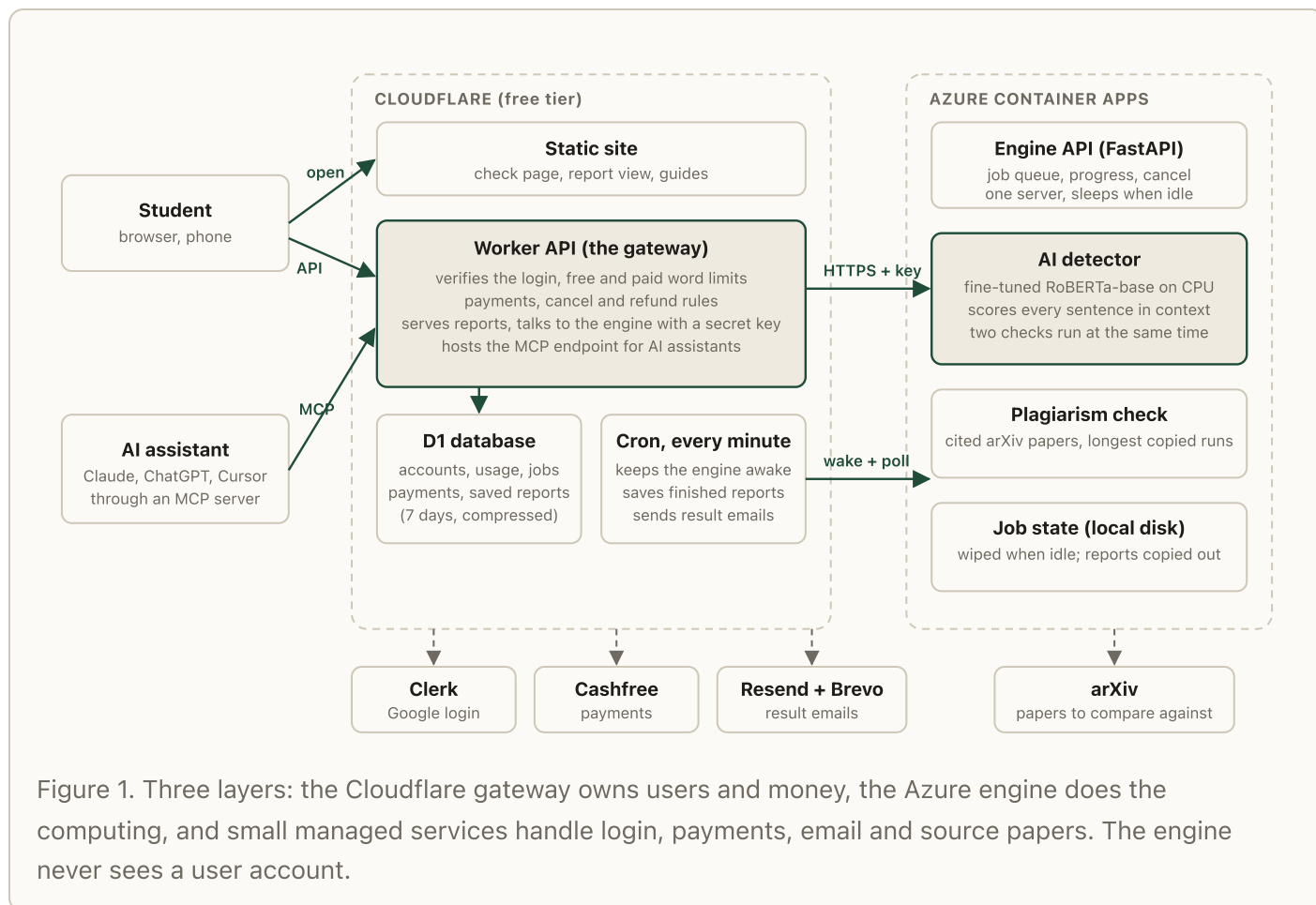
for a typical check
now

3 clouds

Cloudflare, Azure,
GitHub

1 The system at a glance

The site and the money logic run on Cloudflare's free tier. The heavy AI work runs on a separate Python service on Azure that sleeps when nobody is using it. Everything that must survive that sleep (accounts, usage, finished reports) lives in the Cloudflare database, never on the engine.



2 What happens when someone checks a paper

The gateway is built so that nobody is charged until the engine has accepted the job, and every failure path ends in a refund. The page can be closed at any point and the check keeps running.

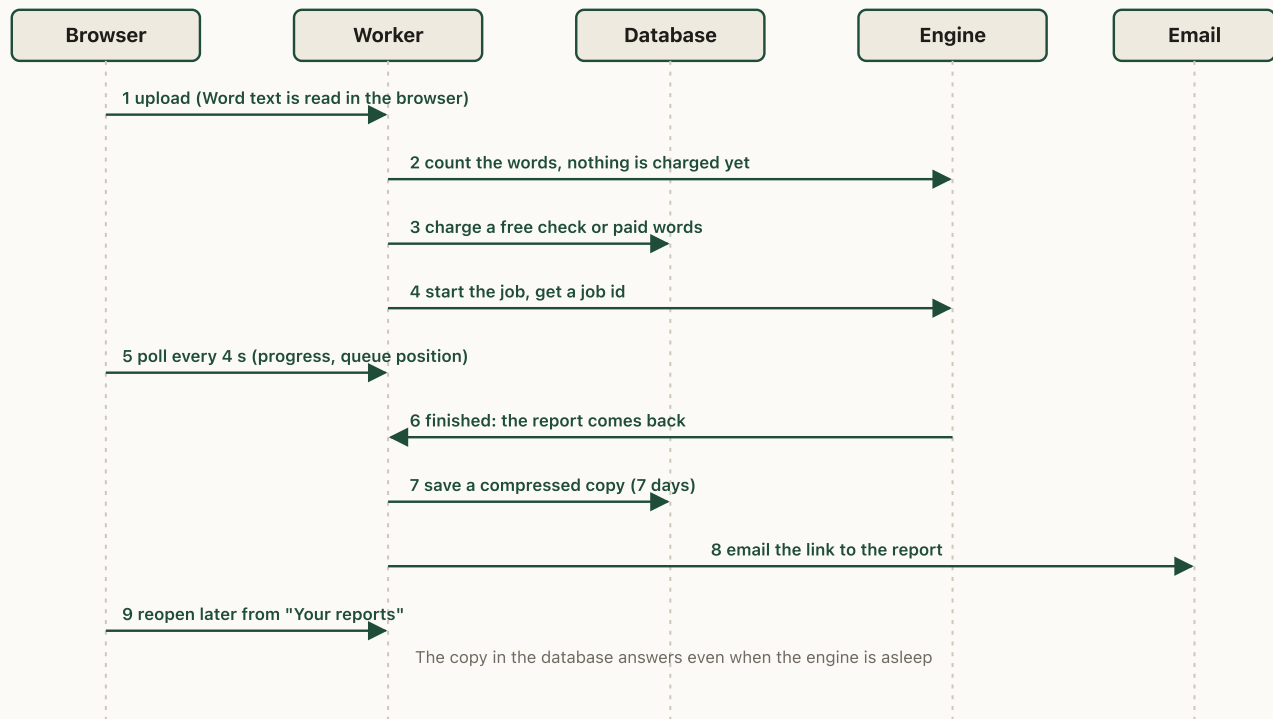


Figure 2. The nine steps. Steps 7 and 9 exist because the engine forgets everything when it sleeps, so finished reports live in the database.

3 How the detector turns text into a number

The score is a measurement of sentences, not a verdict. Each sentence is scored together with the two before and the two after it, because AI-like writing shows up in patterns across sentences, not in a single one.

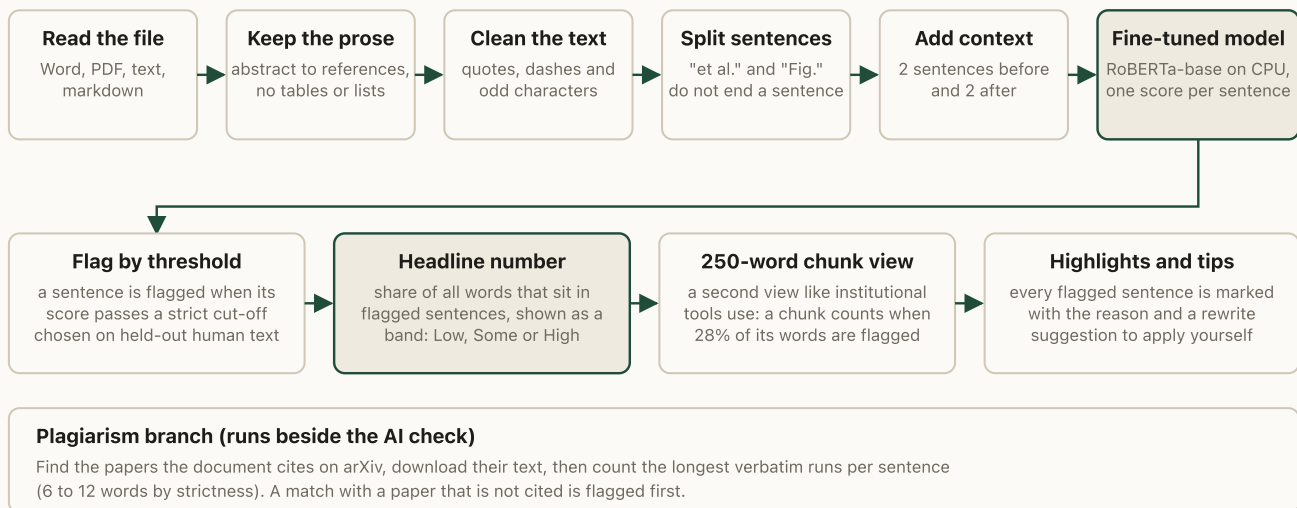


Figure 3. From file to number. The threshold is picked on held-out human writing so that honest text is rarely flagged; the score is shown as an estimate with a band, never as proof.

4 The part nobody sees: money and failure handling

Because the engine can crash, time out or go to sleep, every paid or free check has a clear end state, and each failure state returns the user's words exactly once. A claim row in the database makes the refund safe even if two requests race.

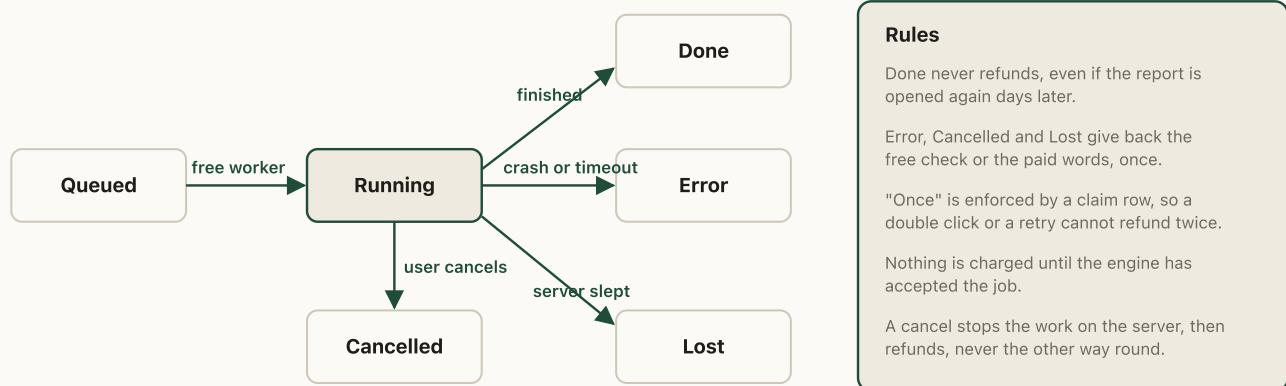
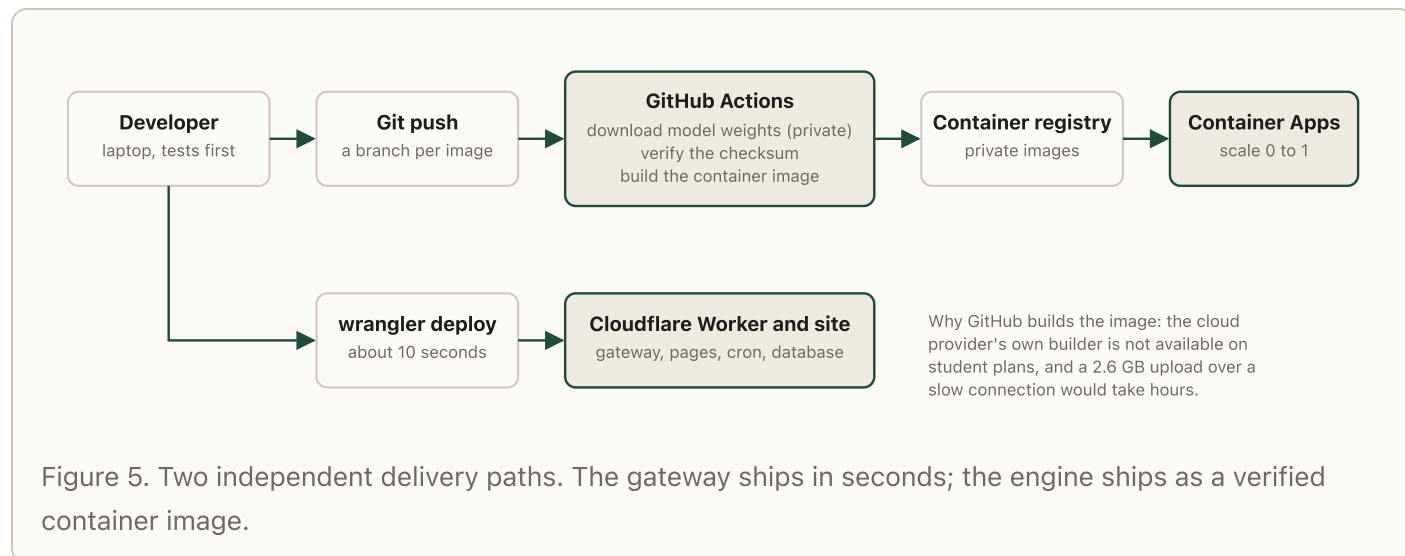


Figure 4. The check lifecycle and its refund rules. These paths were tested with a fake database and a fake engine before launch.

5 How changes reach production



6 What broke, and the numbers behind each fix



Problem	What was really happening	Fix	Effect
Checks took 10 to 60 minutes	Inside the container the AI library used the host machine's core count as its thread count, not the container's 4 vCPUs, so threads fought each other	Set the thread count explicitly	about 10x faster
Four models per check	Only one model set the headline number; the other three added nothing the user saw	Run only that model	about 86% less compute
Queue blocked for an hour	A stuck check held its worker slot until a 1-hour timeout; 21 did in a day	Real timeout, automatic refund	slots freed in minutes
Cost jumped from about 60 to 6,600 rupees a day	Duplicate server versions all scaling up, and a job scanner re-reading about 900 old folders on billed file storage every 30 s	One version, remember finished jobs	file operations down 90%
Database nearly out of free writes	One query rewrote a device row on every poll: 94% of the 100,000 daily writes	Write once, refresh every 6 hours	88% to a few percent
Hosting account switched itself off	The student account has a hard spending cap	Moved the engine to another account; reports saved in the database	identical scores on 341 of 341 sentences
Result emails stopped for hours	A function was called but never defined, so the email loop crashed every minute	Fixed, plus a test that now sends one email per finished check	every check gets an email

7 Design decisions and their trade-offs

Decision	Why	Cost of that choice
Gateway on Cloudflare Workers and D1	Free, fast everywhere, no servers to run, built-in cron	Strict free-tier limits: 100,000 database writes a day, 500 MB, small per-run limits
Separate Python engine	The AI model needs Python and a heavy library; the gateway should stay tiny	Two systems to keep in sync, and a network hop
One fine-tuned model, not an ensemble	Faster, cheaper, and the extra models did not change what users see	Less diversity of opinion; honesty depends on good threshold tuning
Scale to zero, one server only	You pay only while checks run; cold start is about 4 seconds	No spare capacity, and state is local to the one server
Reports saved in the gateway database	The engine's disk is wiped when it sleeps	Reports are stored for 7 days, and storage is compressed to stay in the free tier
Refund claims instead of "if failed then refund"	Retries and double clicks must never pay twice	One extra table row per refund
Scores shown as bands and estimates	Detectors disagree and none is proof	Less dramatic than a single big number, but honest

8 How it was tested

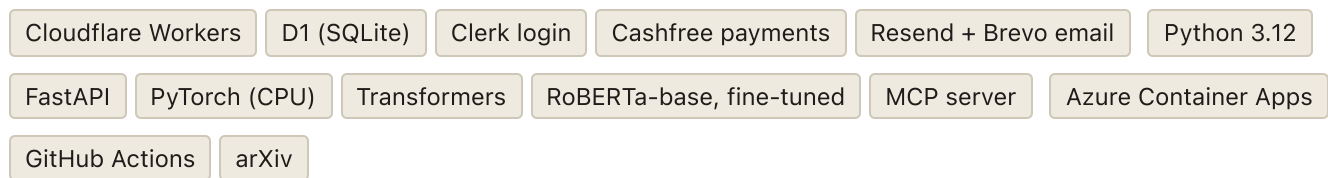
The model

The cut-off was chosen on held-out human writing from several domains (academic papers, encyclopedia text, non-native English writing, learner essays) so honest text is rarely flagged. The target was at most 2 to 3 percent of human documents flagged per domain, and the live setting measured between 0 and 2.1 percent. A held-out set of real student-style AI text checked that detection did not collapse.

The system

34 automated tests run against a fake database and a fake engine, including the dangerous cases: a finished report after the engine sleeps, a lost check refunded once, an old report never refunded, retries that never send two emails. A migration was proven by re-scoring a real production paper on the new host and matching all 341 sentences exactly, and by running four real papers at once.

9 Stack



10 What I would do next

Keep the model loaded between checks instead of reloading it each time (saves 10 to 15 seconds). Try a faster number format (int8) after a proper accuracy check. Move the engine to a permanent free host so a spending cap can never switch it off. Put cost and quota alarms in on day one of the next project.

Honest limits. Scores are estimates, not proof, and different checkers give different numbers. Plagiarism is checked against arXiv only, not the whole web. This is a one-person project and the code and the trained model are private.