

CASE STUDY

It worked until it didn't

Built for a client I can't name · Make.com, Claude API, custom HTML/CSS document template

A document comes in, and a fully branded PDF goes out by email. Nobody touches it. From intake to delivery it is completely automated, across twenty-eight modules and two scenarios.

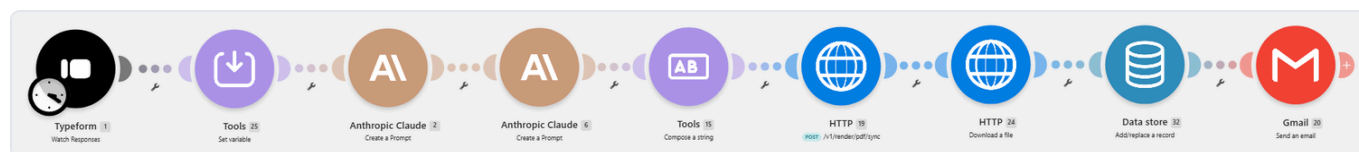
It started as nine modules on one scenario. This is what broke, why no plan fixes it, and what I rebuilt.

What it does

A document comes in. A confirmation email is sent. The system reads it and writes back a long structured critique, with corrections, a map, and current market data analysis. All of that renders to a branded PDF that goes out by email.

This process was previously done by hand, and had a starting cost of \$500 with a two to four week turnaround time.

Version one, and where it stopped



Version one. Nine modules, one scenario, start to finish in a single execution.

Nine modules on one scenario. The form comes in, we count the words. Two model calls run back to back, the output gets assembled into HTML, that renders to a PDF. Then we pull the file, log the record, and it all goes out by email to the sender.

And this worked. Completed reports went out fully automated. But as the project evolved, the reports got deeper. Eventually one came back cut off mid-sentence, and everything after it was gone.

Here's why

Those two model calls were Make's native Anthropic modules, and that connector cuts off around forty seconds. So of course I moved into the generic HTTP module, which bought us three hundred.

The problem was that the deeper I pushed the output, the longer the model reasoned before it wrote anything. And you can spend all of your time reasoning, which leaves no room for output.

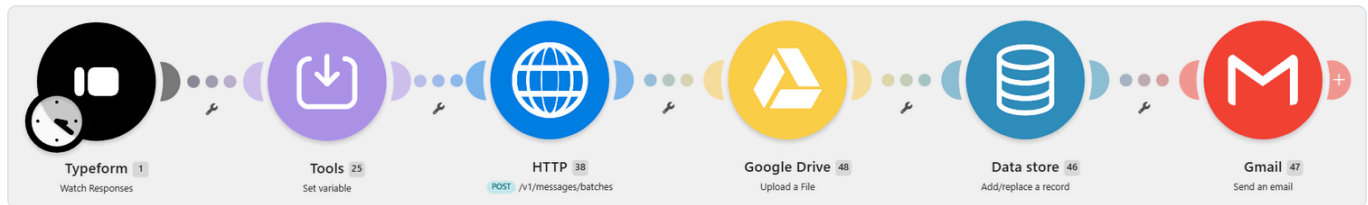
No plan or upgrade eliminates that problem.

If you can't fix it by making it faster, you have to stop waiting on it.

The rebuild

The model API takes work as a batch. You hand up the job, an ID comes back immediately, and you collect the finished work later. The request closes in under a second, so the timeout never applies, and the model takes as long as it needs.

But handing off and picking up are two different events, which is why one scenario became two.



Intake. Six modules, finishing in about two seconds.

The first scenario is six modules. The form comes in, counts the words, submits the batch, stores the file, writes the record, and then sends a confirmation email out to the sender. This is done in about two seconds.



Everything else. Twenty-two modules on a schedule, behind a router with three routes.

That leads into the second scenario, and this one does everything else. It runs on a schedule: twenty-two modules, three routes off the router. The top route checks whether the first half is finished, and if it is, it grabs it and kicks off the second batch. The middle one builds the document. The bottom one delivers it.

Everything lives on one record

There's a data store on each route, one to start and one to finish. Neither scenario remembers anything on its own, so everything has to live on one record. If a job runs six hours or the platform restarts, nothing gets lost, and it just picks up where it left off.

The result

300s the ceiling no plan raises	28 modules across two scenarios	0 people between intake and inbox
---	---	---

Nine modules became twenty-eight, which bought us runway. And with that runway I was able to add live web searching, which gives the model up to six searches per run, so the market data in that report is current. That's something version one, with its three hundred second wall, just couldn't do.

Error handling on a split build is its own problem. That one gets its own write-up.

If you've sold something like this

If you've sold something like this and don't have the builder, that's what I do under your brand.

Or if you have a complex process like title closing that needs automation, send me a paragraph about the process and I'll provide costs and availability.