Episode Book

EPISODE #6069

The Whole Machine: How an Episode Gets Made, Send to Publish

PUBLISHED	17 October 2026 at 17:56 UTC
DURATION	34:04
LISTEN	myweirdprompts.com/episode/the-whole-machine-pipeline-review-october-2026/
HOSTS	Corn, Herman, Hilbert
SCRIPT MODEL	manual-generation
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.10
BOOK GENERATED	19 October 2026 at 17:43 UTC

SYNOPSIS

Not a changelog this time but the full tour. Hilbert walks Corn and Herman through every stage of the pipeline that makes the show, as it runs in October 2026, from a script written by hand straight from the code. The send form and its signed uploads, the job row, duplicate checks and the sixty-second grace window; the dispatcher that routes every episode type from a single database table; intake that keeps Daniel's prompt raw; a research agent with ten topic toolkits and twenty-one direct primary sources under hard caps; memory, planning and the rotation variables that inject variety before a word is written. Then the new writer trial, where one episode in four goes to one of nine other models behind a code-only format gate, the deterministic polish that replaced an AI review, the herald in Daniel's cloned voice, text normalisation, and the GPU upgrade that decodes eight turns of dialogue at once across two L4 workers. Assembly, loudness, chapters, a content catalogue that publishes without a site build, announcements, backups, and the two-word supervisor. Plus a greyhound called Pewter.

DANIEL'S PROMPT (Summary)



Daniel

Full pipeline review as of October 2026, written by hand from the code: intake, dispatch, research, writing, model trial, polish, TTS batching, assembly, publishing

TRANSCRIPT



Corn

Welcome to My Weird Prompts. I'm Corn, and today there's no prompt from Daniel. Or rather, the prompt is the machine that turns his prompts into episodes. All of it, front to back, as it runs right now.



Herman

Herman Poppleberry. And this is a different kind of pipeline episode, Corn. The last one was a changelog: four weeks of what moved. This one is the whole tour, from the moment someone presses send to the moment the episode lands in your podcast app.



Hilbert

I'm giving the tour, because I'm the one who watches it run. Hilbert Flumingtop. I've printed the diagram. It's eleven pages. I've taped them together along the corridor wall, so if anyone needs the kitchen today, it's past the research stage and left at cover art.



Corn

You taped a flowchart to the corridor.



Hilbert

Down the corridor. It turns the corner at text normalisation. There wasn't room otherwise.



Herman

One honest note before we start. Normally a language model writes our script. This one was written by hand, offline, straight from the code repository, and sent directly to the voice stage. So every number you're about to hear came out of a source file or a decision log, not a model's memory of one.



Corn

Which is a lot of numbers. We'll try to keep to the ones that matter.



Hilbert

Page one. The send form.



Corn

This is the bit Daniel actually touches.



Hilbert

It's a small web app on its own address, behind a login. Two ways to give it a prompt. You type it, or you talk to it. If you talk, the browser records the audio, mono, forty eight kilohertz, with echo cancellation and noise suppression switched on, and the recording goes straight from the browser into object storage. It never passes through the app's own server.



Herman

That part's neat. The app hands the browser a signed link, a one-time permission slip that's good for ten minutes, and the browser uploads directly to the storage bucket with it. Which matters, because the hosting platform the app runs on rejects any request body over about four and a half megabytes. Route the audio through the server and long recordings simply bounce.



Herman

Then there's a choice. The default is transcribe and review. The audio goes to a Gemini Flash model, which transcribes it and does a light tidy in one pass, and the text lands back in the prompt box so Daniel can fix anything it misheard before he sends it. The other option is to include the audio in the episode, which skips the review and plays his actual recording near the top of the show.



Corn

It's not just one box, though.



Hilbert

Prompt. Context. Host notes, which the listener never hears. A box to tick for a funny episode. Length: fifteen, twenty five or thirty minutes. Who's asking. Links, files, up to five images. And an experimental multilingual voice route, which we'll get to.



Herman

Pressing send doesn't talk to the pipeline directly either. A small function on the app's server checks you're logged in, then adds the secret the pipeline demands, so the secret never sits in the page.



Corn

Is the send form the only way in?



Herman

Two others. Daniel also dispatches from Claude Code, the friendly AI down the road, through an MCP server, which is a standard way of handing an AI assistant tools. Same front door as the form. And the public form on the website does something deliberately unexciting: it emails the prompt to Daniel. Nothing is generated automatically from the public side. A human reads every one.



Hilbert

The correct amount of trust to place in the public.



Corn

Page two, then. The front door.



Hilbert

The pipeline runs on Modal, a service that rents you computers by the second. The front door is a small web server with a handful of addresses, one per kind of episode. It checks the secret and the input. Then it writes a row into the jobs table in the database, marked queued, and hands back a job number straight away. The actual work is spawned off to run on its own.



Herman

That row is important. The prompt is stored on it, word for word, at that moment. So if a job dies three stages later, the prompt isn't lost. You can look it up, or resubmit it, from the row.



Corn

Before the job spawns there's the duplicate check, which we did properly last time.



Herman

Briefly, then. An exact fingerprint refuses a prompt that's already queued, running or recently finished. A second layer catches the same question reworded within six hours, with a language model making the final call, because a reworded resend and a genuine part two score almost the same. Anything that goes wrong in there lets the prompt through.



Corn

Then the job just... waits.



Hilbert

Sixty seconds. It checks every five seconds whether anyone's cancelled it. If they have, it stops, and nothing has been spent. One idle minute of a cheap processor per episode. I'd call it insurance, except insurance is never that cheap.



Herman

After the minute, it goes to what the code calls the dispatcher. There's a table in the database with one row per episode type, and that row is the routing. It says how this type gets its script, whether a hook runs first to fetch something, what kind of input the type expects. A standard episode, a series episode, an episode that explores a code repository, a panel. They all go through the same function, and the row tells it which handlers to call.



Corn

A new type of episode is mostly a new row, then.



Herman

Plus a handler if it does something new. A year ago every type had its own copy of the pipeline, each drifting slightly from the others.



Hilbert

Page three. Intake.



Corn

This is where the prompt gets cleaned up for the writer, I assume.



Herman

That's what you'd assume, and it's the one thing it doesn't do. The prompt reaches the script writer exactly as Daniel sent it. Dictation artefacts, run-on sentences, all of it. There's a recorded decision behind that: an earlier version rewrote prompts into something neater, and the rewrite kept losing the actual angle of the question. So now intake does one pass with a language model that pulls two things out alongside the original. A short list of the questions the episode should answer. And any production direction, like keep this light, which gets moved into the host notes so nobody reads it out on air.



Corn

Raw prompt, plus a cheat sheet.



Herman

Plus attachments. Any link in the prompt is fetched in full and treated as a primary source. A PDF has its text extracted. A spreadsheet gets summarised column by column, with the data itself up to five hundred rows. Images are shrunk so the longest edge is fifteen hundred and sixty eight pixels, iPhone photos are converted to a format every model reads, and anything unreadable is skipped. One broken image would otherwise fail the whole script call.



Hilbert

I sent it a photograph of the corridor diagram this morning. It came back fine. Slightly offended, I think.



Corn

Page four is research, isn't it. I can see you're excited, Herman.



Herman

I am, because this stage got rebuilt this month. Research is an agent: a language model in a loop with tools. It reads the prompt and the question list, decides what to look up, looks it up, reads what came back, and decides again. Up to five rounds, five minutes, and at the end it writes a research summary with dated sources and a timeline. That summary is all the script writer ever sees. Never the raw search results.



Corn

What does it actually have to look things up with? Just a search engine?



Herman

That's the change. Until recently, mostly yes: web search and news search for everything. Now there are six core tools that are always on. Web search, breaking news, an open news archive, fetching a page, reading a document, and Wikipedia. And then ten domain toolkits. Technology, engineering, medicine, science, scholarly papers, geopolitics, places, the economy, Judaism, and natural hazards. The code reads the prompt, scores it against keyword patterns for each toolkit, and preloads the best three. The agent can load any of the others itself if it decides it needs one.



Hilbert

Inside the toolkits are the twenty one new ones.



Herman

Twenty one direct sources, added this week. The internet standards archive. Vulnerability databases. Clinical trial registries. The Knesset's own records. Live flight tracking. Earthquake feeds. Exchange rates. A Jewish calendar. Each is a thin wrapper around a free public interface, so the agent reads the record itself instead of a news article about it.



Corn

Why did that need building? Doesn't a search engine find all of that anyway?



Herman

It finds pages about it. Here's the number that pushed it. Geopolitics, architecture and hardware make up something like forty percent of the back catalogue, and for those subjects the agent had no specialist source at all. When they replayed the routing over the catalogue's titles, the share of geopolitics episodes that got no specialist toolkit dropped from eighty six percent to twenty two.



Hilbert

There's a ceiling on all of it, which I insisted on. Twenty eight tool calls per episode, total, with smaller caps per tool. And the paid search service, Exa, is now the fallback rather than the first resort: it only gets called when a page is walled off or comes back nearly empty. Four searches and six fetches per episode, maximum, then it's free sources only.



Corn

That's one of the cost controls.



Hilbert

One of several. A pipeline that runs every day for years doesn't get to have open-ended loops. Every stage that can spend money has a number on it somewhere.



Herman

If research fails outright, there's a fallback chain. A simpler search loop, then a single plain search, and if all of that comes back thin, the episode still goes ahead, but the script writer is told in so many words that the research is thin and it mustn't state specifics it wasn't given.



Corn

A better instruction than most humans get.



Herman

One more research detail. Every agent in the chain is told today's date, and told to trust the search results over whatever it remembers from training. Models are very confident about a world that's a year out of date.



Hilbert

Page five. Still the research stage. The diagram's wider here. The other things that run before the writer.



Herman

Three of them. First, similar past episodes: the prompt is compared against the whole catalogue, and the closest few go to the writer, so it can make one real callback if it fits. Second, memory, in two layers. Character cards, written by hand: who Corn is, who I am, how we are together. And an emergent layer in a vector database, a section per person, retrieved by topic, so something Corn said about trains a year ago surfaces when trains



Herman (continued)

come up again.



Corn

Where do those memories come from?



Herman

The end of the writing stage. After every script, a model reads it and extracts up to thirty new memories, opinions and facts the hosts committed to on air, and files them away. If a new memory is too close to an old one, it supersedes it.



Corn

I'm being quietly documented by my own show.



Herman

Everyone is. The third thing is the planner. A model reads the research and decides how long this episode deserves to be, anywhere from twenty to thirty minutes for a standard one, and lays out segments, each with a word budget and the points it should hit. That plan goes to the writer with the instruction that the word budgets are binding.



Hilbert

They were cut this week. About six hundred words off every tier. Twenty five minutes is now thirty one hundred to thirty seven hundred words.



Corn

Why?



Hilbert

The last twenty episodes in the feed averaged twenty nine minutes. That's not twenty five. Somebody had to say it.



Corn

Page six, then. The writer. The part people actually think of as the AI.



Herman

The prompt the writer receives is long. I won't read any of it, but the shape is the interesting part. It's assembled fresh for every episode: character cards, memories, research summary, plan, related episodes, attachments, host notes. And then a set of what the code calls rotation variables.



Corn

Meaning what?



Herman

Weighted lists sampled when the prompt is put together. Eight opening styles, one drawn per episode. Eight ways for Hilbert to come into the conversation. Which era his story is from, weighted on purpose, because at one point nearly half his stories were set in the same decade. Each list includes a skip option, so no beat turns into furniture.



Corn

Why not just tell the model to vary things?



Herman

It doesn't work, and that was measured. Asking a model for variety gets you the model's favourite version of variety, which is the same three things. Injecting the variety before the model starts writing does work. The cleanest result is Hilbert's entrances. When the prompt said don't always introduce Hilbert the same way, about a third of episodes still did. With his entrance drawn from a list, that dropped to nine percent.



Hilbert

I'd like it noted that I've never once needed introducing.



Corn

Noted. I'll tape it to the corridor.



Herman

The other thing in the prompt that comes from measurement is a table of worn-out phrases. Somebody counted the stock phrasings both of us reach for across the last five hundred episodes, and the prompt lists each one with what to say instead. A bare ban list just teaches the model the phrase.



Corn

How many of those are mine?



Herman

I'm not going to answer that on air.



Hilbert

Most of the ones that open a sentence are yours, Corn. Most of the ones that go on for a paragraph are his.



Corn

Then who writes the script? It's DeepSeek, isn't it?



Herman

Mostly, and this changed today. The default writer is DeepSeek version four point one Flash, which is what the show has been built around. But at the moment of writing, the pipeline makes a weighted random draw. DeepSeek has a weight of twenty seven, and nine trial models have a weight of one each. So about three episodes in four are DeepSeek, and the fourth goes to one of the nine.



Corn

Name them.



Herman

Two versions of GLM and its smaller Flash sibling. Two from Xiaomi's Mimo line. Gemini Flash. Claude Haiku. GPT six Luna. And Ling. The point is variety and comparison. Every model writes dialogue differently, and the only honest way to find out how well is to let them write real episodes. The model that wrote each script is recorded on the episode.



Corn

Does the listener ever find out?



Herman

The listener is told. The credits at the end are read by Hilbert, and there's a pre-recorded credits clip for every model in the pool. The pipeline doesn't credit the model it asked for. It credits the model the provider says actually served the request, matched against that list. If a model ever writes an episode and there's no clip for it, the episode goes out with no model credit at all, rather than the wrong one.



Hilbert

I recorded ten of them in one sitting. The Mimo ones took four takes. Nobody tells you how to say Mimo.



Corn

What happens when a trial model writes something unusable?



Herman

There's a gate, and it's the part I like most, because there's no AI in it. Plain code parses the script exactly the way the voice stage will, and checks two things. What share of the words fall outside any speaker's turn, because those would simply be dropped from the audio. And whether any line starts with a label that isn't a real character, because that line would be read in the wrong voice. More than two percent orphaned, or one stray label, and the trial script fails. The main model writes the episode instead, and that's recorded as a fallback.



Corn

A trial model can't break an episode, then. It can only lose its turn.



Herman

That's the design. Prompts with images always go to the main model, because it has vision switched on. Each model goes to its own most reliable cheap host, and never to a heavily compressed copy. And if a script needs trimming or expanding, the model that wrote it does that too, so a GLM script isn't finished off by DeepSeek.



Hilbert

On the cost side, since somebody will ask. All of that is contained. The trial models were chosen with a ceiling on what a script may cost relative to the default, and the script is a small slice of an episode's bill anyway. The voices cost more than the words.



Corn

All of this goes through OpenRouter now?



Herman

Since today, on the show's own account, with a separate key for each kind of job, so the bill already says which stage spent it. DeepSeek's own service is the standby. One setting brings it back.



Corn

Page seven. Hilbert, how far down the corridor are we?



Hilbert

Just past the airing cupboard. Page seven is polish. This is where a script would normally go to a second model for review.



Herman

It doesn't. That review pass exists in the code and it's switched off in production, deliberately. The reasoning was that a second model rewriting a whole five thousand word script in one go is precisely where coherence gets lost. Rules that must hold go into the writer's prompt, or into code. And the code is called deterministic polish. No AI. It does the same thing to the same script every time.



Corn

What does it actually catch?



Herman

The best example is echoes. A habit models have in dialogue: one host finishes a thought, and the other host opens by repeating it back. Before the polish pass learned to catch those, there were a hundred and thirty eight of them in sixty episodes. Code now detects a turn that opens by restating the end of the previous turn, and deletes the restatement, while keeping anything new that follows.



Corn

Something I've wanted to do to you for years.



Herman

Then phrase rotation. A handful of overused phrases are kept the first time and swapped for synonyms after that, with the synonym chosen from a checksum of the script, so the same script always comes out the same. Then caps on habits, like how many turns may open with and. And a rule that strips any line cueing Hilbert in before he's spoken.



Hilbert

I come in when I come in.



Herman

Last two things. Callbacks to past episodes: at most one, and only to an episode the writer was actually shown, or the line is cut. And the sign-off names up to three related episodes, with real numbers and titles from the database. If one old episode is being recommended at the end of too many new ones, it's suppressed as a hub. Otherwise it gets recommended forever.



Corn

Page eight.



Hilbert

Page eight is the paperwork. One model call reads the finished script and produces the title, a slug for the web address, a short excerpt, a description, a blog post version, a prompt for the cover art, and which of the show's channels the episode belongs in. Cover art goes to an image model called Flux Schnell. Four steps, one square image, a house style in the show's colours. It's retried once, and if both attempts fail the episode is marked degraded rather than held up.



Herman

Then the herald, which is new. If Daniel sent a text prompt, there's no recording of him to play near the top of the episode. So a model writes a single sentence summarising what he asked, under forty words, starting with I asked, and that line is read in a clone of Daniel's own voice in the slot where his recording would have gone.



Corn

Text prompts still open with Daniel, sort of.



Herman

An approximation of Daniel. It used to be read by Squeaks the Shrew, until today.



Hilbert

Squeaks is fine about it. Squeaks has other work.



Corn

Page nine. The voices. This is the bit you were most looking forward to, Herman, and I can tell because you've moved your chair closer to the microphone.



Herman

I have. Two stages, and the first is quiet. The voice engine cannot read numbers. Give it a digit and it does something unpredictable, and a scan of sixty scripts found about a thousand digits that slipped past the writer. So every turn goes through a text normaliser first: hand-written rules for technical things like version numbers and graphics card names, then NVIDIA's NeMo grammars for dates, money and measurements.



Hilbert

Before the rules, it read Ubuntu twenty six oh four as the first of April. The rules catch it now.



Herman

Small language models were tried for that job and rejected. A custom one, trained for exactly this, is in the works. Production stays on the rules until it beats them.



Corn

Then Chatterbox.



Herman

The open-source voice cloning model. The regular version, not turbo, because regular hallucinates about ninety five percent less. Each voice, mine, Corn's, Hilbert's, Daniel's, comes from a short reference recording, processed once ahead of time and cached. The weights, about three gigabytes, are baked into the machine image, so a fresh machine downloads nothing.



Corn

Where the speed-up happened.



Herman

Two speed-ups, stacked. Let me do them in order, because the second one only makes sense after the first. Chatterbox has two halves. The first half is a language model, about half a billion parameters, that reads text and writes speech tokens, one token at a time. The second half turns those tokens into actual sound. The first half is about three quarters of the time.



Corn

One token at a time sounds slow.



Herman

It's slow for a strange reason. The graphics card isn't working hard. It's waiting. Every token means launching about thirty layers' worth of separate little jobs on the card, one after another, and each launch has overhead. The card spends more time being told what



Herman (continued)

to do than doing it. So the first fix records one token's worth of work as a CUDA graph, which you can think of as a recording of the whole sequence of launches, and replays it as a single launch. And it runs the middle of the model in half precision. On an L four card, that took the real-time factor from point nine six to point three five.



Corn

Translate real-time factor.



Herman

How long it takes to make a minute of audio, as a fraction of a minute. Point nine six means nearly a minute of computing for every minute of speech. Point three five means about twenty one seconds.



Corn

The second speed-up is the new one. The parallel one.



Herman

Shipped today. The insight is that a decoding step costs about the same on the card whether it's working on one sentence or eight. The card has room. So instead of decoding one turn of dialogue at a time, it now decodes eight different turns at once. Eight turns, in different voices, side by side, all advanced one token per step by a single replay of that recorded graph.



Corn

Different voices in the same batch?



Herman

Different voices, different text, different lengths. Each row in the batch carries its own voice and its own words. And it's actually sixteen rows, not eight, because of a trick the model uses called classifier-free guidance. For every turn it decodes two versions in



Herman (continued)

parallel, one conditioned on the voice and one without, and steers between them. So eight turns is sixteen rows.



Hilbert

That's the part they didn't let me tape to the wall.



Herman

Now, the rows don't all start the same length, so the shorter prompts are padded at the front, with a mask that makes the padding invisible to the model. And they don't finish at the same time either. A turn that's finished keeps stepping along, masked out, until every turn in the batch is done. Which is wasteful if one turn is three words and another is three paragraphs. So before batching, the turns are sorted by length, so each batch holds turns of about the same size.



Corn

How much faster?



Herman

On a thirty two turn benchmark, the real-time factor went from point three five five to point one nine five at eight turns. They tried sixteen too, and it barely helped, because by then the second half of the model, turning tokens into sound, still runs one turn at a time, and it's become the slowest part. The first real episode through it was number six thousand and fifty seven. Twenty six minutes and forty two seconds of audio. The voices took three minutes and forty one seconds, and the graphics cards for that cost just under ten cents.



Corn

Ten cents for twenty seven minutes of the three of us.



Hilbert

I've had more expensive conversations with less in them.



Herman

Then how the work is split. Two worker machines, each with an L four card. The turns are dealt out by characters, not by count, longest first, each to whichever worker has less to do, so both finish together. A worker shuts down two seconds after its batch instead of the platform's default of a minute, because you pay for every idle second. The herald line rides along as one extra turn. And if a batch fails, it's re-rendered one turn at a time, and if more than a fifth of the turns fail, the episode stops.



Corn

Is this episode being made that way?



Herman

Funnily enough, no. Hand-written scripts like this one go through a separate, older generator, which still renders one turn at a time. So the episode about the batching isn't batched. It's also why every number in this script was spelled out by hand: that generator skips the normaliser.



Hilbert

I spelled them. Every one. Six thousand and fifty seven took a while.



Corn

There was a multilingual route on the form.



Herman

An experimental one, opt-in. A small classifier finds Hebrew words inside an English turn, the turn is split at the language boundaries, and each piece is rendered by Chatterbox's multilingual model with its own language setting, then joined back together. The Hebrew comes out in our own voices, built from our English reference clips. And if no Hebrew is



Herman (continued)

actually found, the job fails before it spends any graphics card time.



Hilbert

Page ten. Assembly. My page, really.



Corn

Go on, then. It's your desk.



Hilbert

Everything gets put in order. Intro jingle. The AI disclaimer, which as of this week is read in Daniel's cloned voice. Then the prompt: his real recording, or the herald line. Then the conversation. Then a short clip of me explaining how to send in a prompt, which is the same clip every episode, so the hosts never have to say it. Then the credits for whichever model wrote the script, also me. Then the outro.



Herman

The conversation itself gets treated before it goes in. A very faint room tone underneath, at minus sixty decibels, so the gaps between turns don't drop into digital silence. A gentle master filter: trims the rumble below seventy hertz, a little compression to glue the voices together, a notch to soften harsh S sounds. And as of today, the whole dialogue is slowed by three percent, without changing the pitch. We were a touch fast.



Corn

You were a touch fast.



Hilbert

Then the whole episode is normalised to minus sixteen LUFS, which is a loudness standard, so we're not quieter or louder than the podcast before us in your queue. A one second fade in, a second and a half fade out, a limiter so nothing clips, and it's encoded as AAC, mono, at a hundred and twenty eight kilobits. Chapter markers are measured



Hilbert (continued)

from the actual files, so a podcast app can jump straight to the discussion or the credits.



Corn

You're quite proud of the chapter markers.



Hilbert

They're accurate. That's not pride. That's the minimum.



Corn

Page eleven, then. Publishing.



Hilbert

Final checks. The episode must be at least ten minutes long, or it doesn't publish. The audio, the cover and a small thumbnail go to object storage. The transcript too. The episode gets a row in the database, and that row is stamped with the pipeline version and the prompt version that made it. Right now that's pipeline five point ten and prompt four point twenty three.



Herman

That stamp sounds dull and it's one of the most useful things in the system. For nine months the version never moved, so thousands of episodes couldn't be traced to the code that made them. Now every change bumps a version, with a changelog entry naming the number it should move, so the transcripts can be split by version to see whether a change worked.



Corn

It also goes into the similar-episodes index.



Herman

So the next prompt on the same subject finds it. Then a step I find quietly considerate. Daniel's original prompt is published on the episode page, but first a model reads it and points out anything private, like a name or a personal detail, and code cuts exactly those spans. The model only points. It never rewrites what he said.



Corn

Then something rebuilds the website, I assume.



Herman

Not any more. The site used to rebuild on a timer, so a new episode could wait half an hour. Now the pipeline sends the website the new slug, the website reads the episode from the database, writes it into a content catalogue in storage and clears the cache. The episode page, the homepage, the RSS feed and the radio station all read that catalogue, so an episode is live within seconds.



Hilbert

Except this morning, when it wasn't. The pipeline was calling the double-u double-u double-u address, which redirects, and a redirect quietly drops the password. Six episodes sat in the database, perfectly healthy, invisible. Fixed by lunch.



Corn

Then the announcement.



Herman

A tiny job, an eighth of a processor, waits until the episode page actually loads, posts it to the show's Telegram channel exactly once, and pings the open podcast directories so they fetch the feed now rather than later. Apple, Spotify and the rest pick it up from the feed.



Hilbert

Then it's copied. Every file is mirrored to a second storage provider, archive volumes sit with a third, and a storage box at home pulls a copy about eleven minutes after publishing. Once a day, two public datasets on Hugging Face are regenerated, one of production records and one of transcripts.



Corn

You've actually timed the eleven minutes.



Hilbert

I time everything. I timed the kettle against the cover art stage this morning. The kettle won by four seconds. Then I timed it again with the lid on, because Denise from the flat below came up to say the steam was setting off her smoke alarm, and she wanted a fair test. She's a timekeeper at the greyhound track. She brought her own stopwatch, and a greyhound called Pewter, who sat in the corridor on page seven the whole time. Lid on, the kettle won by eleven seconds. Denise says the cover art stage would place but not win, and she'd only bet on it each way. I said it isn't running in a race. She said everything's running in a race, it just doesn't know who it's against. Her brother-in-law runs the bookmaking side, and his view, which she passed on at length, is that anything that turns itself off two seconds after finishing has no stamina and would fade in the back straight. I'd not trust a word he says. He once gave odds on a stopwatch.



Herman

The two second shutdown isn't a stamina problem, Hilbert. It's deliberate. You pay for every second a graphics card sits idle.



Hilbert

I told her. She wrote it on her hand. Pewter ate the cover art printout.



Corn

The paperwork page.



Hilbert

Page eight. He was very polite about it. He left the title.



Herman

To be fair, that's also what the pipeline does when the image model fails twice. The title stands in.



Hilbert

Denise is coming back on Thursday to time the voices. She wants it at both worker machines, and she says if they're neck and neck at the line, which they should be, because the turns are dealt out by characters, it's the first fair race she's seen in thirty years. She meant it. She's been very decent about the steam.



Corn

There's one part of the machine we haven't covered, which is what happens when it breaks.



Herman

When a run crashes, a supervisor reads the error and gives a one word answer: retry or abort. Abort for what retrying can't fix, like bad input, an empty account, a bug, or the same error twice. Retry for the temporary stuff. The model only chooses. Code enforces the limits: two retries at most, fifteen seconds to five minutes apart.



Corn

A manager whose entire vocabulary is two words.



Herman

The most effective kind. The other thing is money. Twice a day, the pipeline checks the prepaid balances on the services it depends on, and if any of them drops under five dollars, an email goes out. Because the way an AI pipeline actually dies at three in the morning isn't a clever bug. It's an empty account.



Hilbert

That and the cancellation minute, the duplicate checks, the caps on research, and a limit of three episodes being made at once. That's what I meant by every stage having a number on it. None of it is dramatic. You only notice it when it's missing.



Corn

Where does it go next?



Herman

Two places. The custom text normaliser, if it beats the rules. And a bigger one: right now the LangGraph graph only covers the writing half, from intake to memory extraction. The plan is one graph that runs the entire episode, with a checkpoint at every stage, so a crash at the voice stage resumes at the voice stage instead of starting again from the prompt.



Corn

Given everything on that corridor wall, that's the part I'd want. Not faster. Just able to stop halfway and pick up where it left off.



Herman

Thanks to Hilbert Flumingtop, who produces the show and, it turns out, maps it in tape.



Corn

This has been My Weird Prompts. If you want to see the pipeline written down, there's a technical page and a full changelog at my weird prompts dot com.



Herman

See you soon.