Episode Book

EPISODE #6428

Self-Hosting an Analytical Stack Around Postgres

PUBLISHED	3 November 2026 at 19:43 UTC
DURATION	25:53
LISTEN	myweirdprompts.com/episode/self-hosting-analytical-postgres-stack-55818/
HOSTS	Corn, Herman, Hilbert
SCRIPT MODEL	z-ai/glm-5.3
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.26
SOURCES	20 cited by the research (listed at the end)
BOOK GENERATED	4 November 2026 at 09:31 UTC

SYNOPSIS

Running analytical data engines alongside PostgreSQL no longer requires rented cloud warehouses. While permissive open-source tools like ClickHouse, DuckDB, and DuckLake make the query and metadata layers surprisingly lightweight, building a self-hosted stack comes with hidden operational costs. Examining everything from Apache Iceberg catalogs to in-process querying via DuckDB reveals that the analytical engine is simple to self-host, but real-time change data capture pipelines remain full of tricky architectural traps.

TRANSCRIPT



Corn

Five gigabytes of RAM. That was the number that stopped me, because that's the whole reporting stack, Kafka included, in Docker, on five gigabytes.



Herman

And it refreshes hourly. That's the part people don't expect. They hear Kafka and they picture a rack, a team, a budget line. The SolDevelo reference stack runs on less memory than a laptop I owned in the nineties.



Corn

We'll get there. First, the housekeeping. Daniel's finishing his trilogy. Parts one and two covered why you'd pair Postgres with an analytical engine at all, BigQuery, ClickHouse, Snowflake, and then the pipeline between them. Part three is the self-hosting question. His framing is that Postgres is something anyone can run on their own box, while lakes and warehouses feel like weather systems you rent from a cloud provider. So he wants to know which of the open source analytical engines you can run yourself, whether they're real or ornamental, and whether you could assemble the whole thing end to end. Postgres on one side, a self-hosted pipeline in the middle, a self-hosted warehouse on the other.



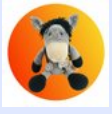
Herman

And the answer splits cleanly in two. The engine side is mature, permissively licensed, and as of this year, dramatically simpler than it was. The pipeline side is where you earn it.



Corn

Which is a nice way of saying the second half of this episode is a warning.



Herman

A friendly warning.



Corn

Those exist?



Herman

Occasionally. The asymmetry Daniel's poking at is real, though. Snowflake and BigQuery are sold as services, so people assume the analytical half of the stack is somehow inherently a service. It isn't. ClickHouse is Apache-2.0. DuckDB is MIT. Iceberg is Apache-2.0. These are not crippleware cores with the real product behind a paywall. ClickHouse's open source repo is the same engine the company sells as ClickHouse Cloud. Same code. Fifty thousand plus stars, pushes literally today, a stable release from last week and an LTS line that lands a day before it.



Corn

The LTS is the tell, isn't it. Hobby projects don't do maintenance branches.



Herman

They don't. And the install is a one-liner, curl the script, pipe it to a shell, done. Monthly release cadence, going back years. They just did a ten year anniversary call for the 26.6 release. This is an adult project with a decade of scars.



Corn

So the engine side of Daniel's question has a short answer. Yes, they exist, and yes, they're serious. Then the interesting question becomes the lakehouse layer, and that's where twenty twenty-six actually changed something.



Herman

It did, and it changed in Postgres's favor, which is why this episode exists as part three rather than a standalone. Two things happened. DuckDB put out version two, Cyanoptera, on October twenty-first. And alongside it, DuckLake one point one.



Corn

Explain DuckLake for the listener who's been ignoring lakehouses on principle.



Herman

Fair instinct. The traditional lakehouse keeps your data as Parquet files on object storage, and then keeps the metadata, which tables exist, which files belong to which snapshot, in one of two places. Either files alongside the data, which gets ugly at scale, or a separate catalog service. Hive Metastore, a REST catalog, Unity Catalog, Nessie. That catalog is a whole extra thing you run and keep alive.



Corn

The piece nobody drew on the diagram until it broke.



Herman

DuckLake's move is to say, no, put all of it in a SQL database. SQLite, DuckDB itself, or PostgreSQL. The data stays Parquet on object storage, the metadata lives in tables in a database you already know how to operate. You get ACID transactions, snapshots, time travel, and there's no catalog server at all. The lakehouse is just... a schema.



Corn

In the Postgres you're already running.



Herman

Which, if you're Daniel and your whole question is about self-hosting a stack around Postgres, is a remarkable gift. The operational database and the lakehouse catalog become the same box.



Corn

It also makes Postgres load-bearing in a way that would've sounded absurd two years ago. If that Postgres is down, it's not just your orders that are down, it's your analytics metadata too.



Herman

True, though I'd argue you were already treating that Postgres as load-bearing. Nobody's casual about their operational database. You're consolidating the blast radius rather than expanding it.



Corn

Mm. I'd want the analytics catalog on a separate instance at minimum. Same engine, separate failure.



Herman

That's a reasonable middle position, and it's one line of config, which is rather the point. One point one landed some thoughtful details. Nested schemas, so you can have lake dot sales dot europe dot orders, proper namespaces. Exact string statistics stored in the catalog, which sounds boring until you realize what it buys you. A query asking for the min or max of a string column can be answered from the catalog table instead of opening Parquet files at all.



Corn

The index becomes the answer.



Herman

The catalog becomes the answer, yes. And row-group counts live in the catalog too, so a reader can plan its scan without fetching every Parquet footer first. If you've ever watched a query spend its first thirty seconds just reading footers on object storage, that detail is not small.



Corn

Adoption signal?



Herman

DuckLake one point zero went production-ready in April. As of last week it's DuckDB's most downloaded lakehouse solution, north of one and a half million downloads in that week alone, with better than two hundred bug fixes merged since one point zero. That's not a research project anymore. That's a thing with a constituency.



Corn

And the migration story?



Herman

Opt-in. The extension reads and writes one point zero catalogs unchanged, and you only migrate by attaching with automatic migration turned on. Nobody got upgraded against their will.



Corn

The one mercy in modern software. Now, Iceberg. The incumbent here.



Herman

Iceberg is the format the big engines standardized on. Born at Netflix, donated to Apache, top-level project, stable spec, a one point twelve release at the end of September. Implementations in Java, Go, Python, Rust, C++. Spark, Flink, Trino, Presto, all speak it. The spec lets multiple engines safely work the same tables at the same time, which is the actual hard problem it solved.



Corn

And its metadata answer is the catalog service.



Herman

Files on object storage plus a catalog. That's the architectural fork in front of anyone building this stack. Iceberg says the catalog is its own service, and in exchange you get an ecosystem where five engines can hit your tables. DuckLake says the catalog is a



Herman (continued)

database, and in exchange you get to not run a catalog service. Iceberg buys you breadth, DuckLake buys you simplicity, and which one wins depends on whether your problem is multiple engines or multiple services.



Corn

For a self-hoster, the second one is the actual problem.



Herman

Almost always. If you're one team with one query engine, you don't need Spark and Trino and Flink all pointed at the same table.



Corn

Before we leave the engines, the other two. DataFusion, and the Postgres-does-everything option.



Herman

DataFusion is the odd one out because it's not a database. It's a Rust library with a full query planner and vectorized columnar execution, plus Ballista if you want to distribute it. You don't run DataFusion, you build with it. It's the route you take when none of the existing shapes fit and you want to assemble a purpose-built engine. Nine and a half thousand stars, very active.



Corn

The engine for people whose answer to buy versus build is always build.



Herman

There's a constituency for that, and honestly it's a good library for it. The other direction is ParadeDB, which says, what second database? It's a Postgres extension stack. One extension does BM25 full-text search, another does columnar analytical processing, and the pitch is that your Postgres becomes the search and analytics engine. No pipeline at



Herman (continued)

all, because there's nothing to pipeline to.



Corn

I notice you sped past that one.



Herman

I did, slightly.



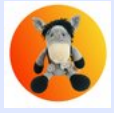
Corn

Because the last time we did the whole is Postgres enough on its own question, it was also me arguing the specialist engine earns its keep. I haven't changed my mind. For dashboards over hundreds of millions of rows, ClickHouse and DuckDB are simply a different weight class, and pretending the extension closes that gap is how people end up rewriting at two in the morning.



Herman

For heavy scanning workloads, agreed. Where ParadeDB is honest is the middle, the shop that has Postgres, has modest analytics, and was about to adopt a second database plus a pipeline for questions their existing box could answer. That's a real population. Though speaking of DuckDB, honestly, with DuckDB being in-process, you could practically run the whole warehouse on a toaster.
Gloria: Clarification, and this is the fourth today. A listener has complained about the phrase a moment ago, and our policy number is seven four one two, on the treatment of appliances. Please do not say a database engine could be run on a toaster. Toasters are not rated for analytical workloads, and that remark was unfair to toasters everywhere. Thank you.



Herman

...Apologies to toasters everywhere. In-process does mean no separate server to manage and no external dependencies, but the toaster will remain dedicated to bread.



Corn

What DuckDB brings to this stack deserves its own minute, though, because of the attach trick.



Herman

The attach trick. DuckDB attaches to PostgreSQL, MySQL, SQLite, natively. Read, join, and move data across engines without a separate ETL step. Your analytical query can sit in DuckDB and join a Parquet file on disk against live Postgres tables. That collapses half of what we called the pipeline in part two. And version two added the quack protocol, where one DuckDB instance becomes an HTTP server other DuckDB clients attach to, so even the sharing story doesn't need a warehouse server.



Corn

Which is the quiet answer to Daniel's question before we've even reached the pipeline. If the analytical engine can read Postgres directly, the pipeline can shrink to almost nothing.



Herman

Almost. And the almost is where the whole rest of this episode lives. DuckDB reading Postgres is great for ad hoc queries. It is not change data capture. If you need the warehouse continuously current, if a row updates in Postgres and the warehouse must reflect it within the minute, attach doesn't do that.



Corn

So, engines ready, lakehouse simplified. Why does the self-hosted stack still feel like a lot of work?



Herman

The pipeline. The canonical self-hosted Postgres to ClickHouse path is Debezium running inside Kafka Connect, writing to Kafka, read out by a ClickHouse Kafka Connect sink, landing in a table, with a materialized view on top. That's four moving parts before you've configured anything.



Corn

And the configuration is where the folklore lives.



Herman

The ClickHouse blog did a good two-part series on exactly this recipe. The interesting parts are what the recipe has to work around. Updates and deletes, mainly. ClickHouse is columnar and append-mostly, so an update from Postgres arrives as a new row and you need the table to know it supersedes the old one. The recipe uses a ReplacingMergeTree keyed on a version column and a deleted flag, so a delete becomes a row with deleted set to true, and the merge engine collapses the history later.



Corn

A delete that's actually a write that pretends to be a delete later.



Herman

Deferred deletion, effectively. And then ordering, which is the part that bites people. Their words, roughly, users should ensure the change events are delivered in order for each unique Postgres row. You get that either with a single partition, the Debezium default, or hash-based partitioning. Get it wrong and the delete can arrive before the update it was supposed to end.



Corn

So the hard parts are exactly-once semantics, delete handling, ordering per row, schema drift across the boundary, and keeping backfill consistent with the stream. None of which are query speed.



Herman

None. And that's the misconception worth killing, because every marketing page leads with benchmark numbers and the benchmarks were never the risk. ClickHouse and DuckDB are both fast. Self-hosted stacks don't die of slowness. They die of the consumer group that silently stopped three weeks ago.



Corn

There's a Herman sentence I want to hold onto. But first, the uncomfortable bit. That blog series we just cited, it carries a notice now.



Herman

It does. I'll paraphrase rather than read it, but the gist is, the content is from twenty twenty-three, many approaches remain valid, and for current guidance they recommend ClickPipes, which is ClickHouse Cloud's managed ingestion, for Postgres CDC. The vendor's own documentation now routes you away from the open source recipe toward the paid product.



Corn

The engine's free and the recipe's behind the till.



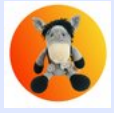
Herman

That's the shape of it. The license isn't the self-hosting tax. Apache-2.0 costs you nothing. The tax is that the operational knowledge, the blog posts, the walkthroughs, the here's how you actually wire Debezium to this, is being written for the managed product first. The company raised three hundred fifty million in Series C funding last year at a six and a third billion valuation. ClickHouse Cloud is the thing that money expects to grow.



Corn

And there's no self-hosted equivalent of ClickPipes waiting in the wings.



Herman

I haven't seen one. The open source answer is still the twenty twenty-three recipe, the exact recipe the vendor now flags as dated. That's a strange position. The engine is a first-class piece of software, and the path to feeding it from Postgres is a blog post the vendor has quietly retired.



Corn

Meanwhile the reference stacks that do exist are all assembled by hand. Walk me through the SolDevelo one.



Herman

Postgres as the source, Debezium CDC through Kafka Connect, Kafka in KRaft mode, so no zookeeper at least, into the analytics store, and then dbt Core for transformation, Airflow for orchestration, Superset for dashboards. Airflow refreshes the marts hourly by default. The lot fits in about five gigabytes of RAM in Docker.



Corn

Five gigs. My laptop has four fans and it still sounds like takeoff.



Herman

It's modest, and it's a real, documented, working stack. But look at what it is. Six services. Postgres, Debezium, Kafka, the sink, Airflow, Superset, plus dbt in the middle. Nobody ships this as one installable thing. It's a Docker Compose recipe on a repository. LumiBase documents a similar shape, Kafka broker, Debezium connector, ClickHouse sink, an Airbyte connector, all on a shared network in Compose. Every complete self-hosted stack I can find is an assembly, four to six services, documented rather than packaged.



Corn

Which answers part of Daniel's question in a way he may not have wanted answered. You can assemble it. Nobody will sell it to you assembled.



Herman

Nobody. And the market's had years to. The absence is informative. Either it's easier to operate as a managed product and the assembly is the moat, or nobody's found the packaging that people would trust. Probably both.



Corn

Are there lighter paths than Debezium and Kafka?



Herman

A few. Sequin is an Elixir, MIT-licensed Postgres CDC service that pushes changes to Kafka, SQS, Elasticsearch, or plain HTTP endpoints, and the notable part is no Kafka Connect required. One service instead of a connect framework plus a broker. If your analytical store is just a second Postgres, there's Kuvasz-Streamer, Go, does Postgres to Postgres consolidation and CDC. Small project, but pointed at a real niche.



Corn

And underneath all of it, the question neither of those answers. Do you need CDC at all?



Herman

The honest question. If the warehouse can be fifteen minutes stale, a periodic export is a script. Copy out of Postgres, write Parquet, done. If it can be a day stale, it's a cron job. Debezium plus Kafka is a lot of machinery to buy for freshness nobody asked for. I've watched teams install the full CDC stack for a dashboard the CEO opens on Mondays.



Corn

That's the inverse toaster. Enormous appliance, one slice a week.



Herman

So the realistic stack comes in two shapes, and I'd put them to Daniel as a fork. Shape one, the SolDevelo shape. Postgres, Debezium, Kafka or Redpanda, the ClickHouse sink, ClickHouse on the far side, dbt Core, Airflow, Superset. Six services, hourly marts, five gigs of RAM, and you own every piece of it, including the ordering guarantees. Shape two, the DuckLake shape. Postgres as the operational database and the lakehouse catalog, DuckDB as the query engine, Parquet on object storage as the data, and the ingestion path is whatever you can live with. Maybe just periodic exports, maybe attach when you need live reads. Two to three things to operate instead of six.



Corn

Two bets about how much machinery a person should own.



Herman

And the second bet only became available this year. In April, DuckLake one point zero. Last week's download numbers. If that bet is right, the self-hosted analytical stack collapses toward Postgres plus a query engine, and the pipeline, the part we've spent half this episode warning about, shrinks to a decision about freshness rather than an architecture.



Corn

If it's wrong, Iceberg's catalog-service model keeps the machinery, and the assembly tax stays. My guess, and it's only a guess, is both survive, because they're not really competing. Iceberg is for the multi-engine world, DuckLake is for everyone else.



Herman

I'd sign that. The one thing I'd add, and it tempers my own enthusiasm, is that we've been talking architecture and adoption signals all episode and I can't hand you a measured self-hosted versus managed throughput comparison. DataFusion's own readme points at ClickHouse's benchmark site for performance claims. The engines being fast is well established. The self-hosted versus managed comparison, on operations rather than queries, is a gap nobody's published. It's a gap worth noticing, because it's exactly the number a team making this decision wants.



Corn

Does anyone actually run the Debezium recipe in production though, as their real pipeline, not a demo?



Herman

People do run it, but you're right that the documented examples skew toward demos and blog posts, and the production ones tend to be...



Hilbert

Wexler runs it. Has done for a year and a half. And Herman, the part you left out of the recipe is the connector config, which is where all of it actually lives. Debezium doesn't come correct out of the box for that sink. I spent an evening on the snapshot mode settings alone. Wexler wanted his depot stock on a dashboard, that was the entire brief, and the working config ended up on an index card in my wallet, because I wasn't typing it twice.



Corn

Wexler being?



Hilbert

Car parts, over on the industrial side. Sparks plugs, brake rotors, pallets of the stuff. Nice man, terrible handwriting. I built him Postgres, Debezium, Kafka, ClickHouse, the lot, exactly the stack you two have been circling for twenty minutes. Herman would've enjoyed the connector tuning. Corn, you'd have hated the first week.



Corn

I'd have hated it by the second day.



Hilbert

While you were on about consumer groups stopping quietly, I rang him. Topic lag was four messages, which is where it always sits on a Tuesday. He reads me the number off the Grafana panel himself now, he's got the phone balanced on his shoulder while he's serving a customer. Doesn't need me. Sold him the little server along with the config, it's his outright, and I've got nothing of that stack left except the index card, which stays in the wallet.



Herman

Four messages of lag on a live CDC feed, sustained, that's a healthy deployment by any standard I'd use in the clinic.



Hilbert

It never moved off four. That's how I knew it was right. Steady number, no surprises, and he pays the electric bill on it, not me.



Corn

The index card is doing more version control than half the repositories we've discussed.



Hilbert

It's survived two wallets. It'll outlast the card format itself.



Herman

One thing, though. If the lag ever sits at exactly zero for hours on a depot that's open, that's the number I wouldn't trust. No movement on a Tuesday morning means something upstream stopped, not that the stockroom went still.



Hilbert

...That had occurred to me. I check the source table now too, not just the panel. Same habit you two have got me into with your logs and your five-point checklists.



Herman

So the honest close on Daniel's question. The engines are ready, the lakehouse got simpler this year, and the pipeline is where the assembly tax lives, and it's a tax paid in services you run and recipes nobody maintains for you, not in licenses.



Corn

And the open question the whole field turns on is DuckLake's bet. If the catalog really belongs in the database you already operate, the stack drops from six services to two or three, and Postgres stops being just the operational database and becomes the spine of the entire business. If that bet's wrong, the machinery stays, and the docs keep drifting toward the managed side. One of those is a bet about software. The other's a bet about incentives, and I know which one I'd rather have my money on.



Herman

Credit where it belongs to our producer, Hilbert Flumingtop, who apparently now runs the only production Debezium stack we've ever personally verified.



Corn

If you want more of this, try episode two, Local STT For AMD GPU Owners; episode twelve thirty-three, Why Just Use Postgres Isn't Always Enough; and episode three, Safetensors or something else. This has been My Weird Prompts. If you've been with us for the whole Postgres run, drop a review and tell us which stack you'd bet on, the six services or the two.



Herman

Find everything at my weird prompts dot com. We'll be back soon.



Corn

See you then.

SOURCES

What the research for this episode read before the script was written. Primary sources first.

1. **last push 2026-11-03, v26.10.2.103-stable (2026-10-24) [primary]**
   https://github.com/ClickHouse/ClickHouse

2. **last push 2026-11-03, v2.0.0 Cyanoptera (2026-10-21) [primary]**
   https://github.com/duckdb/duckdb

3. **last push 2026-11-03 [primary]**
   https://github.com/apache/datafusion

4. **apache-iceberg-1.12.0 (2026-09-30) [primary]**
   https://github.com/apache/iceberg

5. **DuckLake 1.1 announcement, 2026-10-21 [primary]**
   https://ducklake.select/2026/10/21/ducklake-11/

6. **DuckLake 1.0 announcement, 2026-04-13 [primary]**
   https://ducklake.select/2026/04/13/ducklake-10/

7. **DuckLake extension docs, DuckDB 2.0 [primary]**
https://duckdb.org/docs/2.0/core_extensions/ducklake

8. **DuckLake repo [primary]**
https://github.com/duckdb/ducklake

9. **ClickHouse blog, content from 2023, with 2026 staleness notice**
https://clickhouse.com/blog/clickhouse-postgresql-change-data-capture-cdc-part-2

10. **ClickHouse blog, 2023**
https://clickhouse.com/blog/clickhouse-postgresql-change-data-capture-cdc-part-1

11. **reference self-hosted reporting stack**
https://github.com/SolDevelo/soldevelo-reporting-stack

12. **LumiBase Docker Compose CDC deployment docs**
https://docs.lumibase.dev/en/docs/cdc/deployment-docker-compose/

13. **Postgres CDC, MIT, 2,224 stars**
https://github.com/sequinstream/sequin

14. **Postgres-to-Postgres CDC, AGPL-3.0**
https://github.com/kuvasz-io/kuvasz-streamer

15. **archived Debezium/Postgres/Redpanda/ClickHouse demo**
https://github.com/514-labs/debezium-cdc/

16. **last edited 2026-11-01**
https://en.wikipedia.org/wiki/ClickHouse

17. **last edited 2026-08-20**
https://en.wikipedia.org/wiki/Apache_Iceberg

18. **last edited 2026-10-03**
https://en.wikipedia.org/wiki/Unity_Catalog

19. **InfoQ coverage of DuckLake 1.0, May 2026**
https://www.infoq.com/news/2026/05/ducklake-sql-catalog/

20. **ParadeDB profile (pg_search + pg_analytics)**
https://github.com/api-evangelist/paradedb