



# Supercharge your Postgres with Extensions

Adam Hendel <[adam@tembo.io](mailto:adam@tembo.io)>, @adamhendel



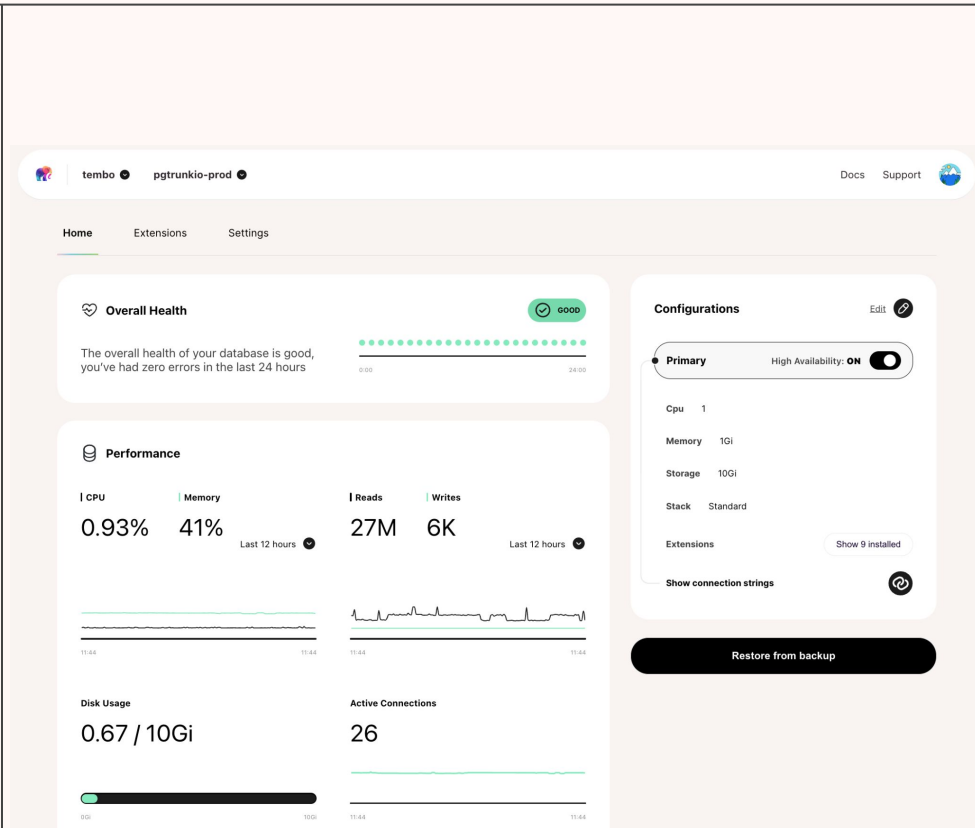
# About me

- Founding Engineer at Tembo
- 10+ years as Postgres user
- Backend engineer; analytics, data science, machine learning
- Author of `pgmq`, `pg_later`, `pg_vectorize`



# Tembo

- Fully-managed database as a service
- 8 variations of Postgres 'Stacks'
- Toggle 190+ extensions on and off
- Reduce complexity of data ecosystem
- Developer-first, fully-extensible Postgres



# Agenda

- 01 Extensions
- 02 Enhancing Postgres with Extensions
- 03 Replacing other data stores with extended Postgres
- 04 Extended Postgres recipes
- 05 Conclusion

# Extensions

- Augment Postgres with new functionality
- Packaged and distributed separately from Postgres
- Can be added onto the database without changing/recompiling it
- Add new data types, functions, optimizer, index and table methods



# Extensions Landscape

Range in complexity from simple new data types to distributed databases

[1000s of Postgres extensions](#)

Varying levels of visibility, maturity, usage.

Smaller number are actively used

## auto\_explain

The auto\_explain module provides a means for logging execution plans of slow statements automatical...

Auditing / Logging Featured

## pgmq

A lightweight distributed message queue. Like AWS SQS and RSMQ, on Postgres.

Featured Orchestration

## pg\_partman

Extension to manage partitioned tables by time or ID.

Orchestration Featured

## pg\_prewarm

The pg\_prewarm module provides a convenient way to load relation data into either the operating sys...

Data / Transformations Featured

## pg\_stat\_statements

The pg\_stat\_statements module provides a means for tracking planning and execution statistics of al...

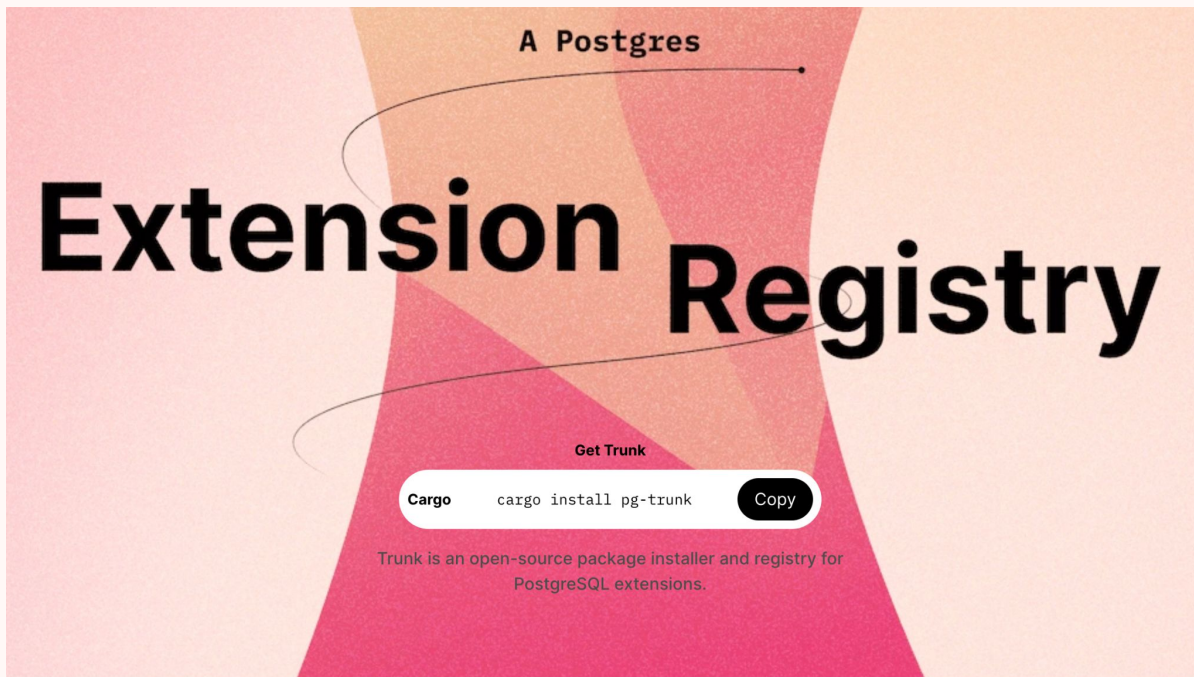
Metrics Featured

## pgvector

Open-source vector similarity search for Postgres

Machine Learning Featured

# Trunk - pgt.dev



A Postgres

# Extension Registry

Get Trunk

**Cargo** cargo install pg-trunk **Copy**

Trunk is an open-source package installer and registry for PostgreSQL extensions.

The banner features a background of overlapping, semi-transparent geometric shapes in shades of pink, red, and orange. The text is centered and uses a clean, sans-serif font. A thin black line with an arrow points from the text 'A Postgres' to the top of the 'Registry' part of the title. Below the title, there is a 'Get Trunk' section with a code snippet and a 'Copy' button. At the bottom, a short descriptive sentence is provided.

tembo

Featured

7

Analytics

13

Auditing / Logging

7

Change Data Capture

6

Connectors

24

Data / Transformations

45

Debugging

2

Index / Table Optimizations

14

Machine Learning

4

Metrics

17

Orchestration

9

## All Extensions

Search 192 extensions

Search

### pg\_partman

Extension to manage partitioned tables by time or ID.

Featured

Orchestration

### pg\_prewarm

The pg\_prewarm module provides a convenient way to load relation data into either the operating sys...

Data / Transformations

Featured

### pg\_stat\_statements

The pg\_stat\_statements module provides a means for tracking planning and execution statistics of al...

Featured

Metrics

### pgvector

Open-source vector similarity search for Postgres

Featured

Machine Learning





# Enhancing Postgres With extensions



## New Data types and functions

### citext

Use a case insensitive character string type.

```
SELECT 'a'::citext = 'A'::citext AS
t;
t
---
t
(1 row)
```

### pg\_uuidv7

Create valid version 7 UUIDs in Postgres.

```
SELECT uuid_generate_v7();
          uuid_generate_v7
-----
-
018570bb-4a7d-7c7e-8df4-6d47afd8c8fc
(1 row)
```

### ltree

Store data in a hierarchical tree-like structure.

```
SELECT count(*) FROM tree WHERE 'A' @>
path;
count
-----
7
(1 row)
```

# Efficient approximate analytics functions

## t-digest

Efficient percentiles for parallel apps / OLAP index.

```
INSERT INTO t SELECT random() FROM
generate_series(1,10000);
SELECT tdigest_percentile(c, 100,
0.95) FROM t;
   tdigest_percentile
-----
 0.9513706724006539
(1 row)
```

## hll

Estimate distinct values with high accuracy.

```
INSERT INTO hll_example (hll_set)
SELECT
hll_add_agg(hll_hash_integer(i)) FROM
generate_series(1, 10000) i;
SELECT hll_cardinality(hll_set) FROM
hll_example;
   hll_cardinality
-----
 9725.852733707077
(1 row)
```

## topN

Set rule-based top value returns.

```
INSERT INTO events VALUES ('a'),
('b'), ('a'), ('b'), ('a');
SELECT (topn(topn_add_agg(name),
2)).* FROM events GROUP BY name;
 item | frequency
-----+-----
  a   |          3
  b   |          2
(2 rows)
```

# Indexes

## bloom

Create indexes based on bloom filters.

```
CREATE INDEX bloomidx ON bloom_test
USING bloom (i1,i2,i3) WITH
(length=80, col1=2, col2=2, col3=4);
EXPLAIN ANALYZE SELECT * FROM
bloom_test WHERE i1 = 500000 AND i2 =
300000;
```

## hypopg

Test if creating an index could be useful without doing it.

```
SELECT * FROM
hypopg_create_index('CREATE INDEX ON
hypo (id)');
```

## rum

Enhance GIN ranking, phrase search, & timestamp ordering.

```
INSERT INTO test_array VALUES ('{1,2,3,4}');
CREATE INDEX idx_array ON test_array USING
rum (i rum_anyarray_ops);
SELECT * FROM test_array WHERE i && '{1}'
ORDER BY i <=> '{1}' ASC;
```

# Efficient monitoring and debugging

## pg\_buffercache

Inspect buffer cache state.

```
SELECT * FROM
pg_buffercache_summary();
- [ RECORD 1 ]---+-----
buffers_used   | 248
buffers_unused | 2096904
buffers_dirty  | 39
buffers_pinned | 0
usagecount_avg | 3.141129
```

## pg\_wait\_sampling

Collect sampling-based statistics on wait events.

```
-- review all the current wait events
SELECT * FROM
pg_wait_sampling_current;
-- to filter the view for a single
process, run
SELECT * FROM
pg_wait_sampling_get_current(pid);
-- recent wait events
SELECT * FROM
pg_wait_sampling_history;
```

## pg\_stat\_statements

Track statistics of SQL planning and execution.

```
SELECT.. FROM pg_stat_statements
ORDER BY total_exec_time DESC LIMIT
5;
- [ RECORD 1 ]---+-----
query          | UPDATE ..
calls          | 3000
total_exec_time | 25565.855387
rows           | 3000
hit_percent    | 100.000000000000
```

# Procedural Languages

## plrust

Rust procedural language.

```
CREATE FUNCTION add_two_numbers(a
NUMERIC, b NUMERIC) RETURNS NUMERIC
STRICT LANGUAGE plrust AS $$
Ok(Some(a + b)) $$;
SELECT add_two_numbers(2, 2);
```

```
add_two_numbers
-----
4
```

## plv8

Javascript procedural language.

```
DO $$ plv8.eblog(NOTICE, 'this', 'is', 'inline',
'code'); $$ LANGUAGE plv8;
```

-- produce a notice log message

## plpython

Python procedural language.

```
CREATE FUNCTION pymax (a integer, b
integer) RETURNS integer LANGUAGE
plpython3u AS $$ return max(a, b) $$;
SELECT pymax(4, 5);
```

```
pymax
-----
5
(1 row)
```

## Connect to other data stores - foreign data wrappers

### mysql\_fdw

Integrate with relational MySQL databases.

```
CREATE SERVER mysql_server
FOREIGN DATA WRAPPER mysql_fdw
OPTIONS (host '', port '');
...
CREATE FOREIGN TABLE ...
SERVER mysql_server
OPTIONS (dbname '', table_name '');
```

### mongo\_fdw

Integrate with NoSQL MongoDB databases.

```
CREATE SERVER mongo_server
FOREIGN DATA WRAPPER mongo_fdw
OPTIONS (address '', port '');
...
CREATE FOREIGN TABLE ...
SERVER mongo_server
OPTIONS (database '', collection '');
```

### postgres\_fdw

Integrate with external PostgreSQL servers.

```
CREATE SERVER foreign_server
FOREIGN DATA WRAPPER postgres_fdw
OPTIONS (host '', port '', dbname
'); ...
CREATE FOREIGN TABLE foreign_table...
SERVER foreign_server
OPTIONS (schema_name '', table_name
');
```

# Automate and orchestrate

## pg\_cron

Schedule cron-based jobs.

```
SELECT cron.schedule(  
  'nightly-vacuum', '0 10 * * *',  
  'VACUUM');
```

```
schedule  
-----  
43
```

## pg\_later

Execute SQL now and get the results later.

```
SELECT pglater.exec(  
  'SELECT * FROM  
  pg_available_extensions order by name  
  limit 2' );
```

```
job_id  
-----  
1  
(1 row)
```

## pg\_partman

Create and manage table partition sets.

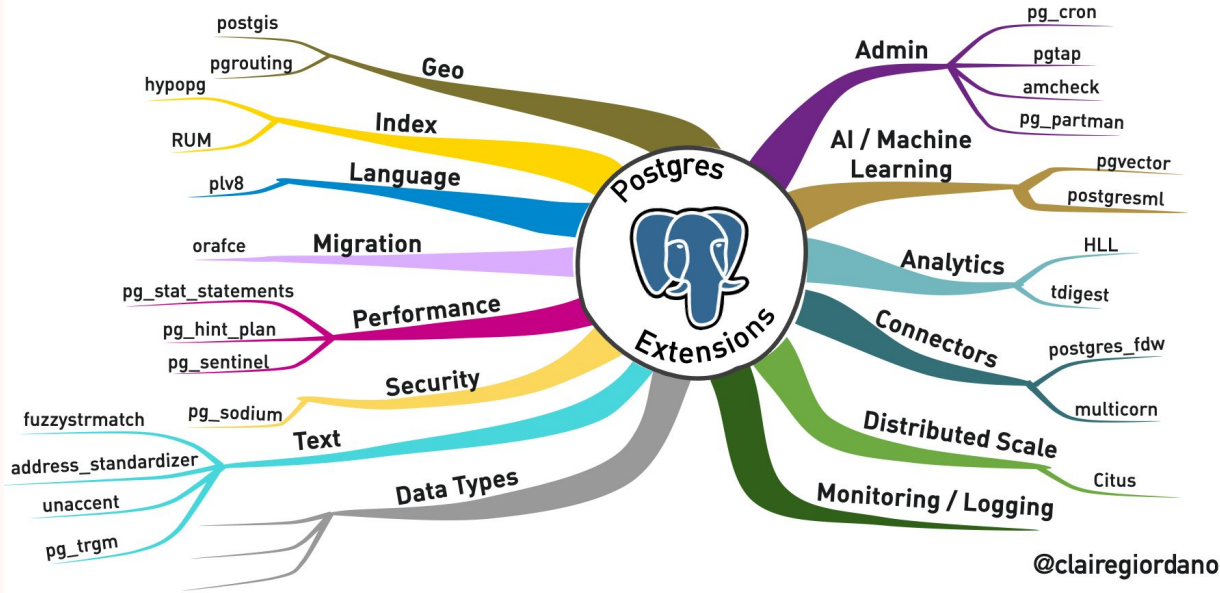
```
SELECT partman.create_parent(  
  p_parent_table :=  
  'partman_test.id_taptest_table',  
  p_control := 'col1',  
  p_interval := '10');
```

```
SELECT  
partman.run_maintenance('public.origi  
nal_table');
```



# More extensions

Claire's work-in-progress "mind map" of PG extensions

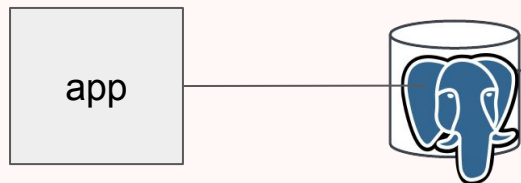




# Replacing other data stores with Postgres



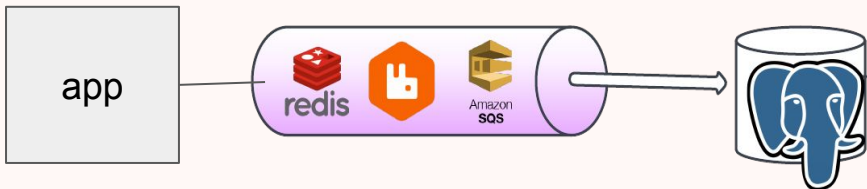
# Modern Data Stack



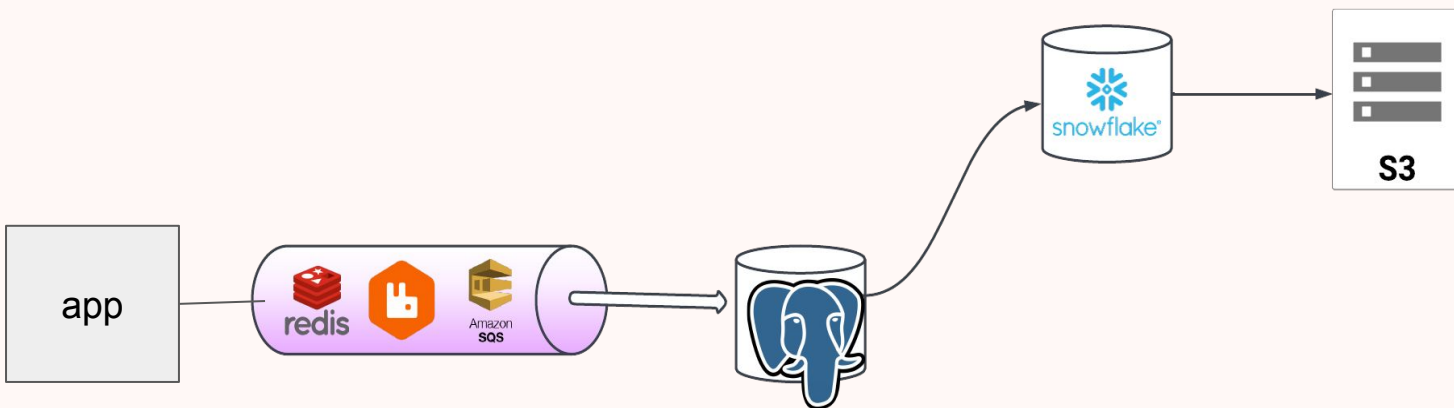
tembo

# Modern Data Stack

tembo

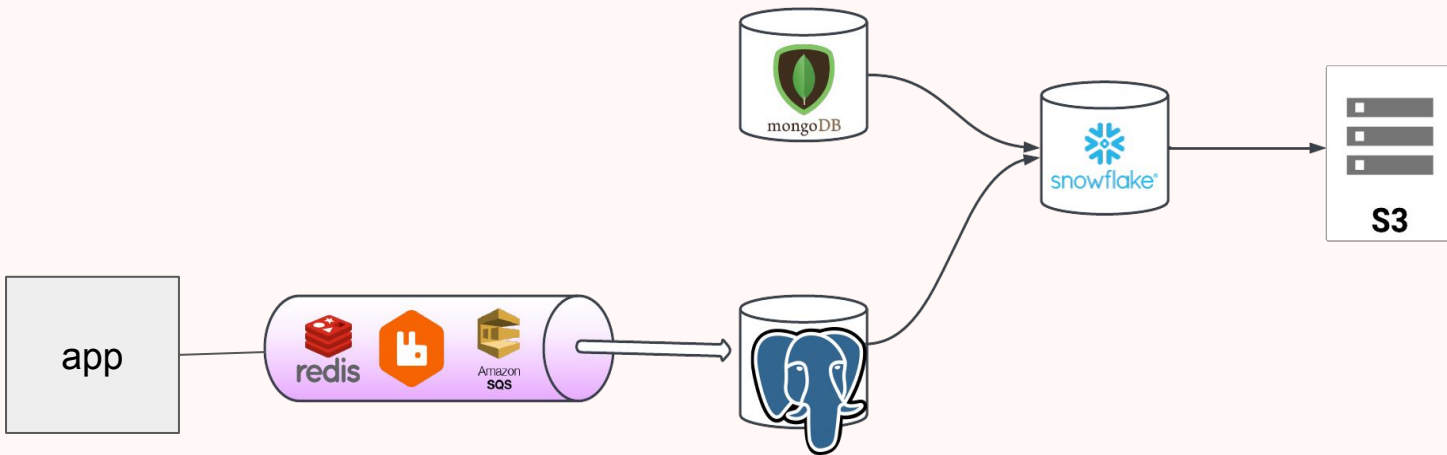


# Modern Data Stack



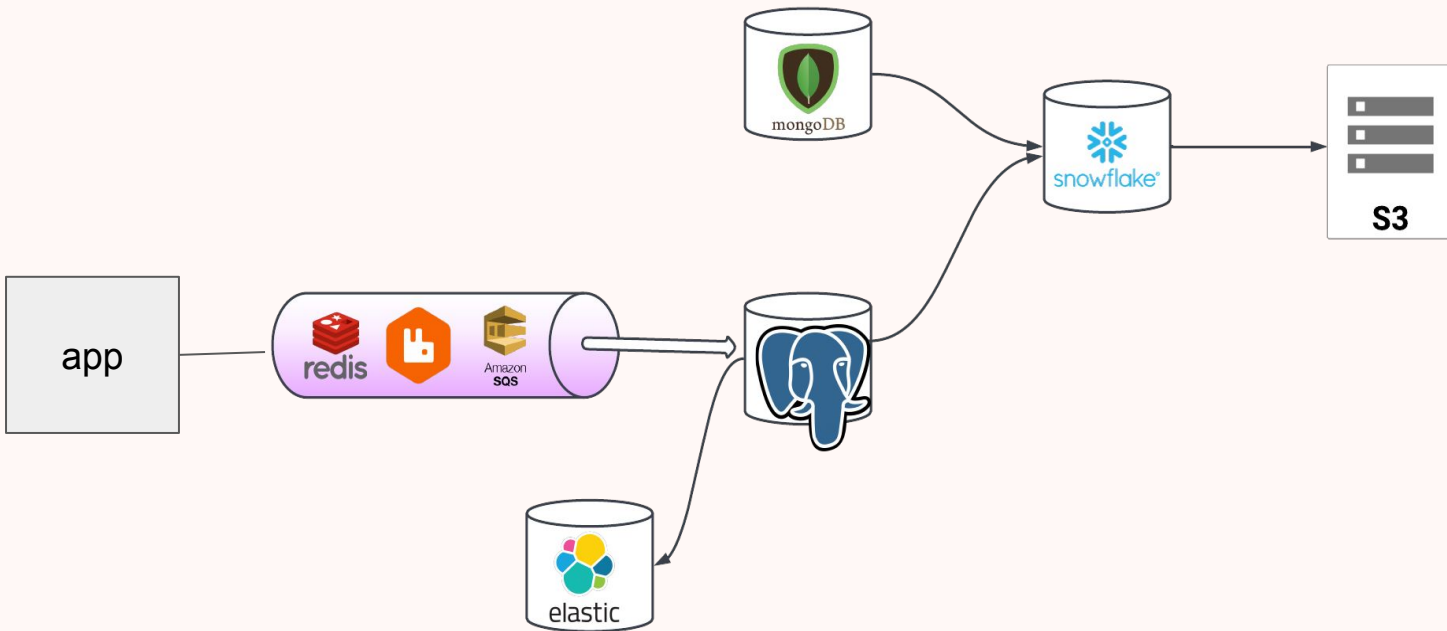
tembo

# Modern Data Stack



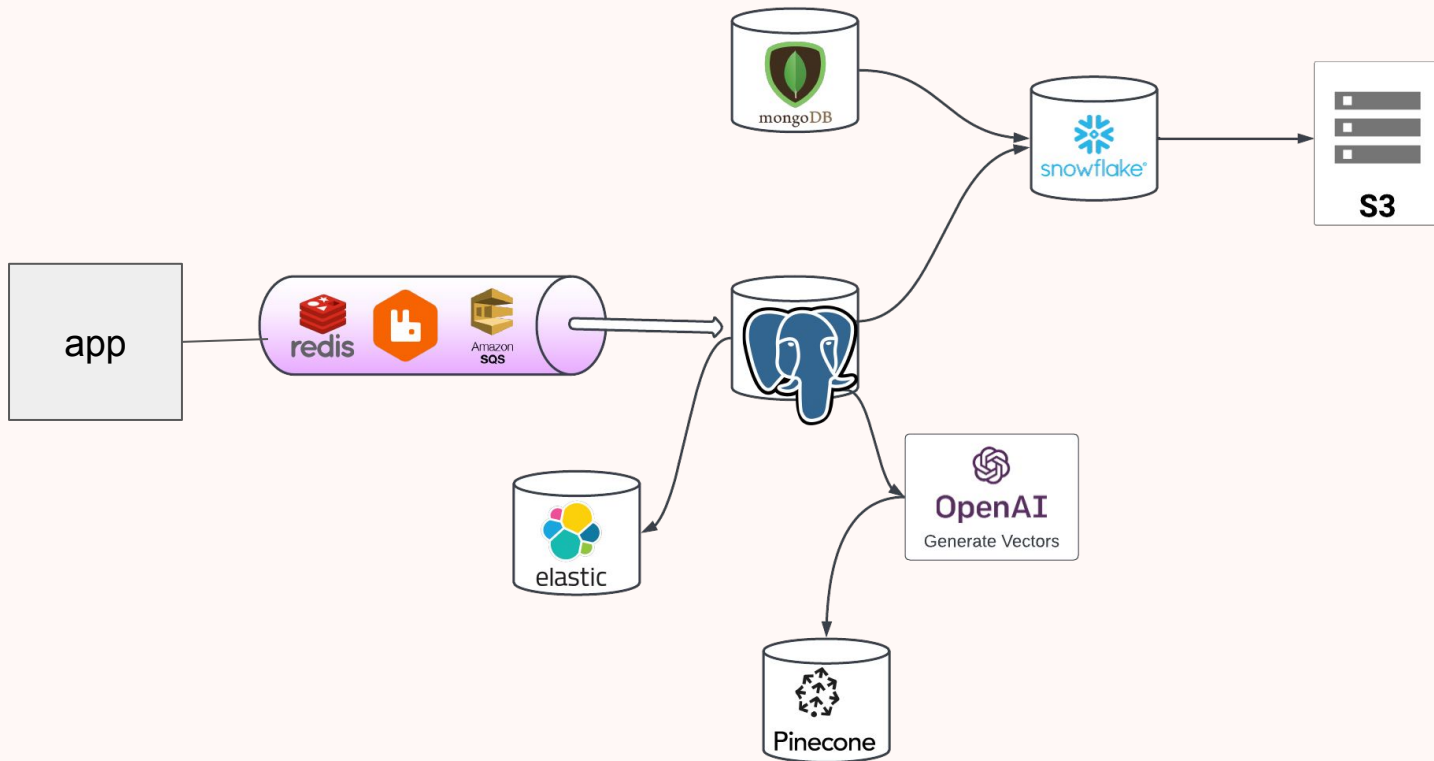
tembo

# Modern Data Stack



tembo

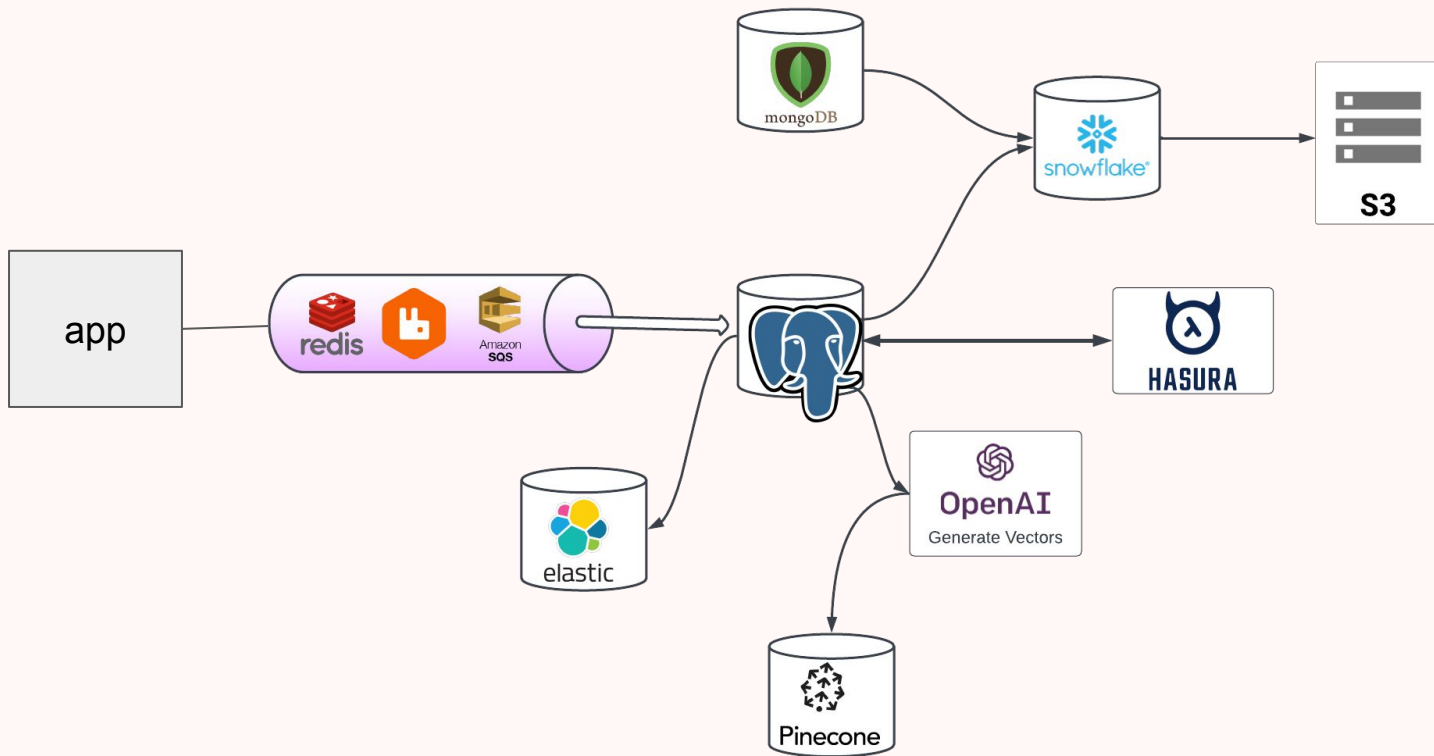
# Modern Data Stack



o  
p  
m  
e  
t



# Modern Data Stack



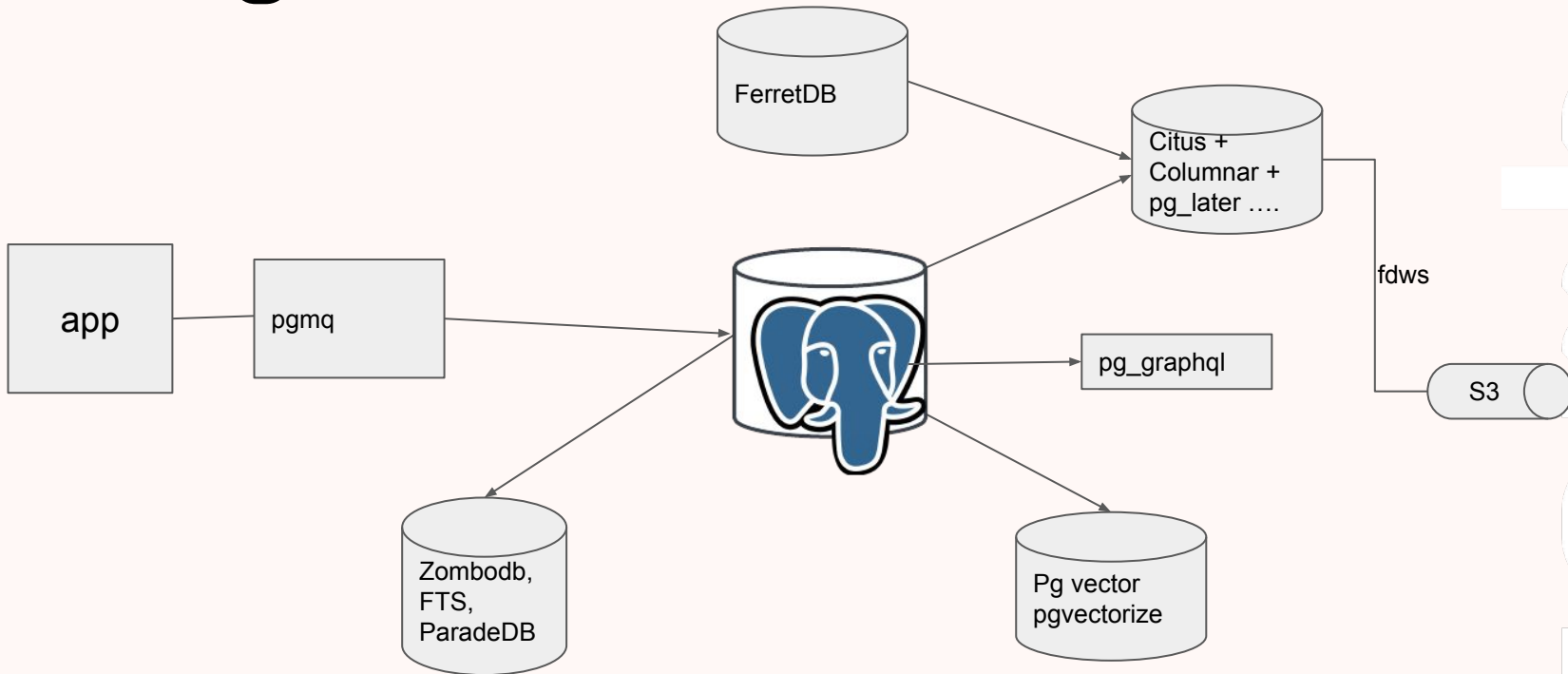
of  
e  
f

# Do you need this complexity?

- Complicated to set up, understand and debug
- Hard to manage, require special skills sets for each
- Piece together ETL tools for each, keep data in sync
- Deal with so many different vendors
- Cost and complexity grows quadratically

o  
r  
m  
e  
t

# Postgres with extensions



# Use Extensified Postgres

Incumbent	Postgres based substitute
Snowflake	Columnar, citus, foreign data wrappers, pg_later and more
Pinecone	Pg_vector, lantern, pg_vectorize
MongoDB	FerretDB
Elastic Search	Zombodb, paradedb
SQS / RabbitMQ	pgmq

o  
r  
m  
e  
t

# Why not?

- Extensions aren't known
- Different maintainers
- Different maturity levels
- Recipes are unknown
- Multiple Postgres for X companies





# Workload Optimized Postgres



# Stacks



CREATE A NEW INSTANCE

## Select a Stack

Exit 

### Geospatial

Select

Postgres with all your geospatial needs.

Extensions:	postgis
Configuration:	Optimized for geospatial workloads
Compute:	Use case based memory and CPU compute profiles

### VectorDB

Select

Postgres tuned and configured for embedding vectors. Generate, store, and search vector embeddings with ease.

Extensions:	pg_vector, pg_vectorize, pg_cron, pgmq
Configuration:	Optimized for memory intensive workloads
Compute:	Use case based memory and CPU compute profiles

### OLTP

Select

Deploy a low latency transactional processing stack to ensure that database transactions with high I/O performance, concurrency, and real-time metrics/observability.

Extensions:	(7) extensions: specialized workload, monitoring, and connection
Configuration:	Tuned for optimized WAL and autovacuum settings, low latency, connection pooling, and indexing
Compute:	Balanced memory and CPU compute profiles

### OLAP

Select

Postgres tuned for online analytical processing. Deploy a stack that is optimized for large data sets, complex queries, and high throughput.

Extensions:	(7) extensions: specialized workload, monitoring, and connection
Configuration:	Tuned for high memory usage, storage, parallelism, and query optimization
Compute:	Balanced memory and CPU compute profiles

### Machine Learning

Select

Deploy Postgres configured for machine learning training and inference workloads in your database. Build your application on PostgresML, pg\_vector and pg\_vectorize for an experience like Sagemaker and Pinecone.

Extensions:	(5) extensions: specialized data type, feature, embedding, & online transformers
Configuration:	Optimized for compute-heavy workloads
Compute:	Use case based memory and CPU compute profiles

### Message Queue

Select

Deploy a Tembo Stack tuned and configured for message and job queue workloads. Build your application using the PGMQ extension for a similar experience to AWS SQS and Redis Simple Message Queues.

Extensions:	(5) extensions: specialized data type, feature, embedding, & online transformers
Configuration:	Optimized and tuned for low latency
Compute:	Use case based memory and CPU compute profiles

### Standard

Select

A Postgres instance balanced for general purpose computing. You have full control over compute, configuration, and extension installation.

Extensions:	None
Configuration:	Recommended defaults
Compute:	Use case based memory and CPU compute profiles

### DataWarehouse

Select

Postgres tuned and configured for datawarehouse workloads. Extract, Transform and Load data from external sources using extensions. Build centralized datastore for analytical and tactical queries.

Extensions:	columnar, multicorn2, wrappers, parquet_s3_fdw
Configuration:	Optimized for batch inserts and memory intensive workloads
Compute:	Use case based memory and CPU compute profiles

# Anatomy of a Stack

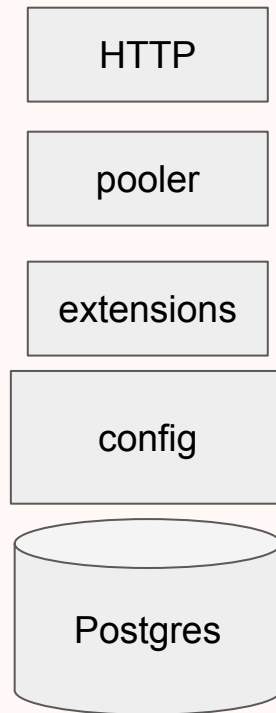
- Docker Base Image containing Postgres
- Curated set of extensions which turn Postgres into best-in-class for that workload.
- Hardware profile recommendations - namely: CPU, Mem, Storage
- Postgres configs optimized according to hardware and workload
- Use-case specific metrics and alerts
- Application deployment add-ons - Deploy a containerized application near Postgres to expand capabilities while minimizing network latency





# Message Queue

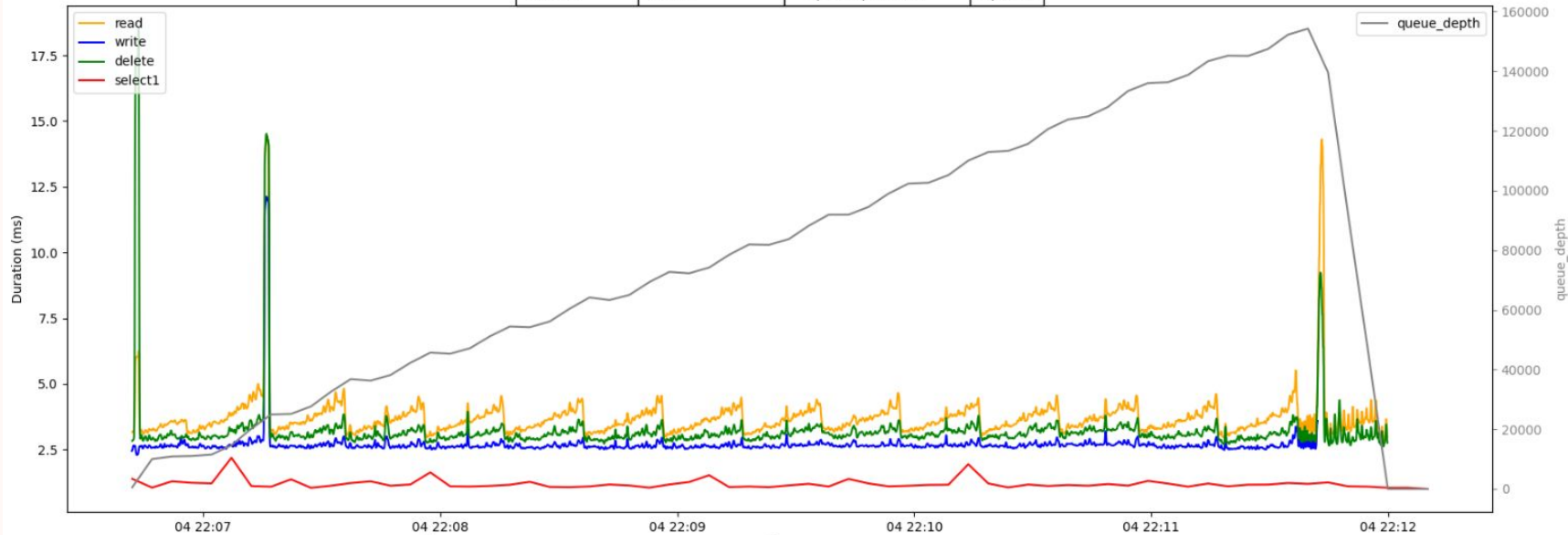
- HTTP interface to queue operations - PostgREST
- Server side connection pooling - pg bouncer
- Extensions:
  - pgmq
  - pg\_partman
- Postgres Configurations
  - Aggressive autovacuum
  - Shared\_buffers
  - Others
- Open Source Postgres



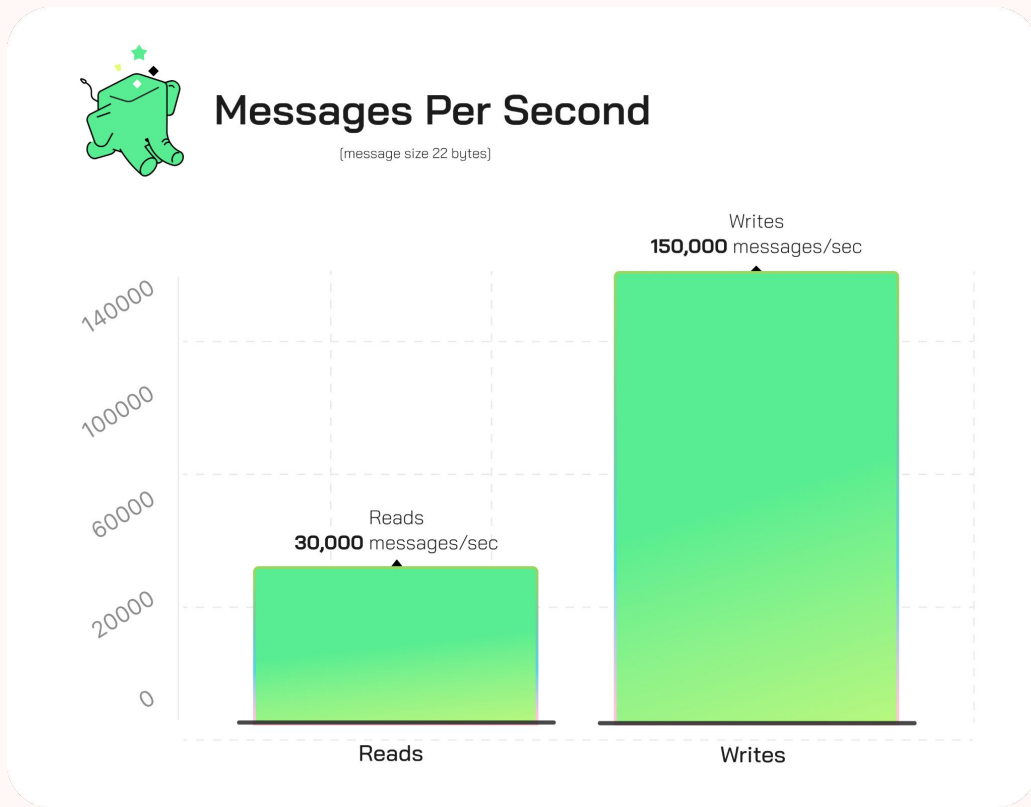
o  
o  
m  
e  
t

# Benchmarking ~25k ops / sec

Operation	Duration (s)	Total Messages	msg/s
delete	317	2,812,242.0	8,852
read	317	2,812,242.0	8,852
write	300	2,812,242.0	9,371

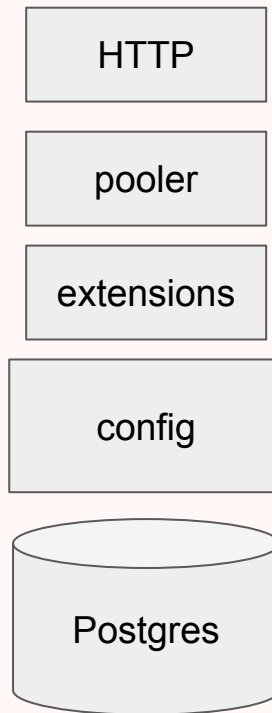


# Tiny messages, large batches



# Vector DB

- HTTP interface to queue operations - PostgREST
- Server side connection pooling - pg bouncer
- Extensions:
  - pg\_vectorize
  - pgvector
  - pgmq
  - pg\_cron
- Postgres Configurations
  - parallel\_workers
  - shared\_buffers
  - others
- Open Source Postgres



o  
a  
m  
e  
t

# Managed Embeddings w/ pg\_vectorize

product_id	product_name	description	last_updated_at
1	Pencil	Utensil used for writing and often works best on paper	2023-07-26 17:20:43.639351-05
2	Laptop Stand	Elevated platform for laptops, enhancing ergonomics	2023-07-26 17:20:43.639351-05

```
SELECT vectorize.table(  
  job_name => 'product_search',  
  "table" => 'products',  
  primary_key => 'product_id',  
  columns => ARRAY[  
    'product_name', 'description'  
  ]  
);
```

```
SELECT * FROM vectorize.search(  
  job_name => 'product_search',  
  query => 'accessories for mobile devices',  
  return_columns => ARRAY['product_id',  
    'product_name'],  
  num_results => 3  
);
```

search\_results

```
{"product_id": 13, "product_name": "Phone Charger", "similarity_score": 0.8564774308489237}  
{"product_id": 24, "product_name": "Tablet Holder", "similarity_score": 0.8295404213393001}  
{"product_id": 4, "product_name": "Bluetooth Speaker", "similarity_score": 0.8248579643539758}
```



### Geospatial

[Select](#)

Postgres with all your geospatial needs.

Extensions:	postgis
Configuration:	Optimized for geospatial workloads
Compute:	Use case based memory and CPU compute profiles



### VectorDB

[Select](#)

Postgres tuned and configured for embedding vectors. Generate, store, and search vector embeddings with ease.

Extensions:	pg_vector, pg_vectorize, pg_cron, pgmq
Configuration:	Optimized for memory intensive workloads
Compute:	Use case based memory and CPU compute profiles



### OLTP

[Select](#)

Deploy a low latency transactional processing stack to ensure that database transactions with high I/O performance, concurrency, and real-time metrics/observability.

Extensions:	(7) extensions: specialized workload, monitoring, and connection
Configuration:	Tuned for optimized WAL and autovacuum settings, low latency, connection pooling, and indexing
Compute:	Balanced memory and CPU compute profiles



### OLAP

[Select](#)

Postgres tuned for online analytical processing. Deploy a stack that is optimized for large data sets, complex queries, and high throughput.

Extensions:	(7) extensions: specialized workload, monitoring, and connection
Configuration:	Tuned for high memory usage, storage, parallelism, and query optimization
Compute:	Balanced memory and CPU compute profiles



### Machine Learning

[Select](#)

Deploy Postgres configured for machine learning training and inference workloads in your database. Build your application on PostgresML, pg\_vector and pg\_vectorize for an experience like Sagemaker and Pinecone.

Extensions:	(5) extensions: specialized data type, feature, embedding, & online transformers
Configuration:	Optimized for compute-heavy workloads
Compute:	Use case based memory and CPU compute profiles



### Message Queue

[Select](#)

Deploy a Tembo Stack tuned and configured for message and job queue workloads. Build your application using the PGMQ extension for a similar experience to AWS SQS and Redis Simple Message Queues.

Extensions:	(5) extensions: specialized data type, feature, embedding, & online transformers
Configuration:	Optimized and tuned for low latency
Compute:	Use case based memory and CPU compute profiles



### Standard

[Select](#)

A Postgres instance balanced for general purpose computing. You have full control over compute, configuration, and extension installation.

Extensions:	None
Configuration:	Recommended defaults
Compute:	Use case based memory and CPU compute profiles



### DataWarehouse

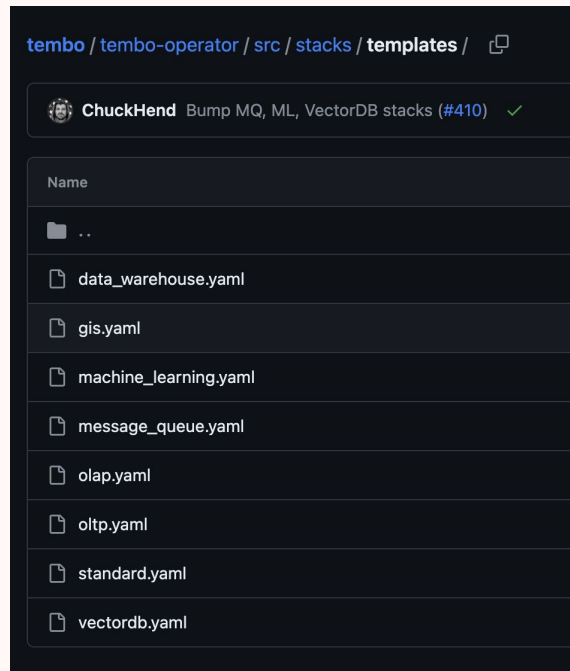
[Select](#)

Postgres tuned and configured for datawarehouse workloads. Extract, Transform and Load data from external sources using extensions. Build centralize datastore for analytical and tactical queries.

Extensions:	columnar, multicore2, wrappers, parquet_s3_fdw
Configuration:	Optimized for batch inserts and memory intensive workloads
Compute:	Use case based memory and CPU compute profiles

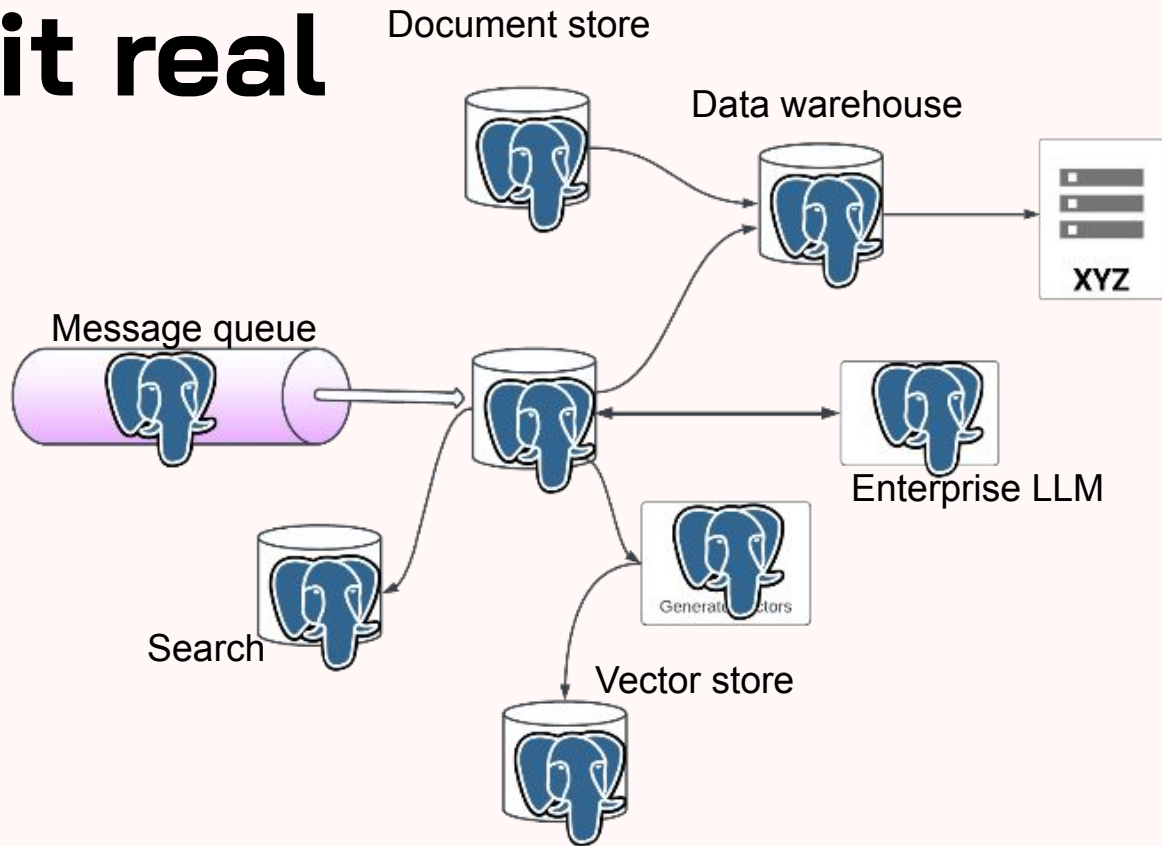
# Open-source recipes

<https://github.com/tembo-io/tembo/tree/main/tembo-operator/src/stacks/templates>



Open-Source Recipes

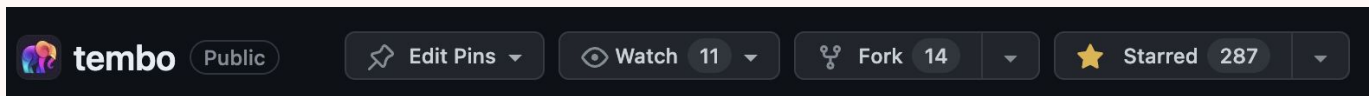
# Making it real



Open  
Source  
AI



# Star us and Contribute



- All these recipes are open source at <https://github.com/tembo-io/tembo>
- Can be run locally with this [guide](#)
- Deployed with a single click on [cloud.tembo.io](https://cloud.tembo.io)
- Open PRs with suggestions
- Help us make them better!

# Extensions FTW!

- Extremely powerful add-ons to Postgres
- Improving extension discoverability, maturity and documentation is important
- Can help you optimize your current Postgres usage
- Can be used to transform Postgres into something else
- Stacks are pre-built recipes for flavored, use-case optimized Postgres
- Try extensions or stacks of your choice on Tembo Cloud





# Thank you! Questions?

Email me at [adam@tembo.io](mailto:adam@tembo.io)

Tweet to us at [@adamhendel](https://twitter.com/adamhendel), [@tembo\\_io](https://twitter.com/tembo_io)

