Processing Trillions of Records with Tiny Serverless Databases

about me

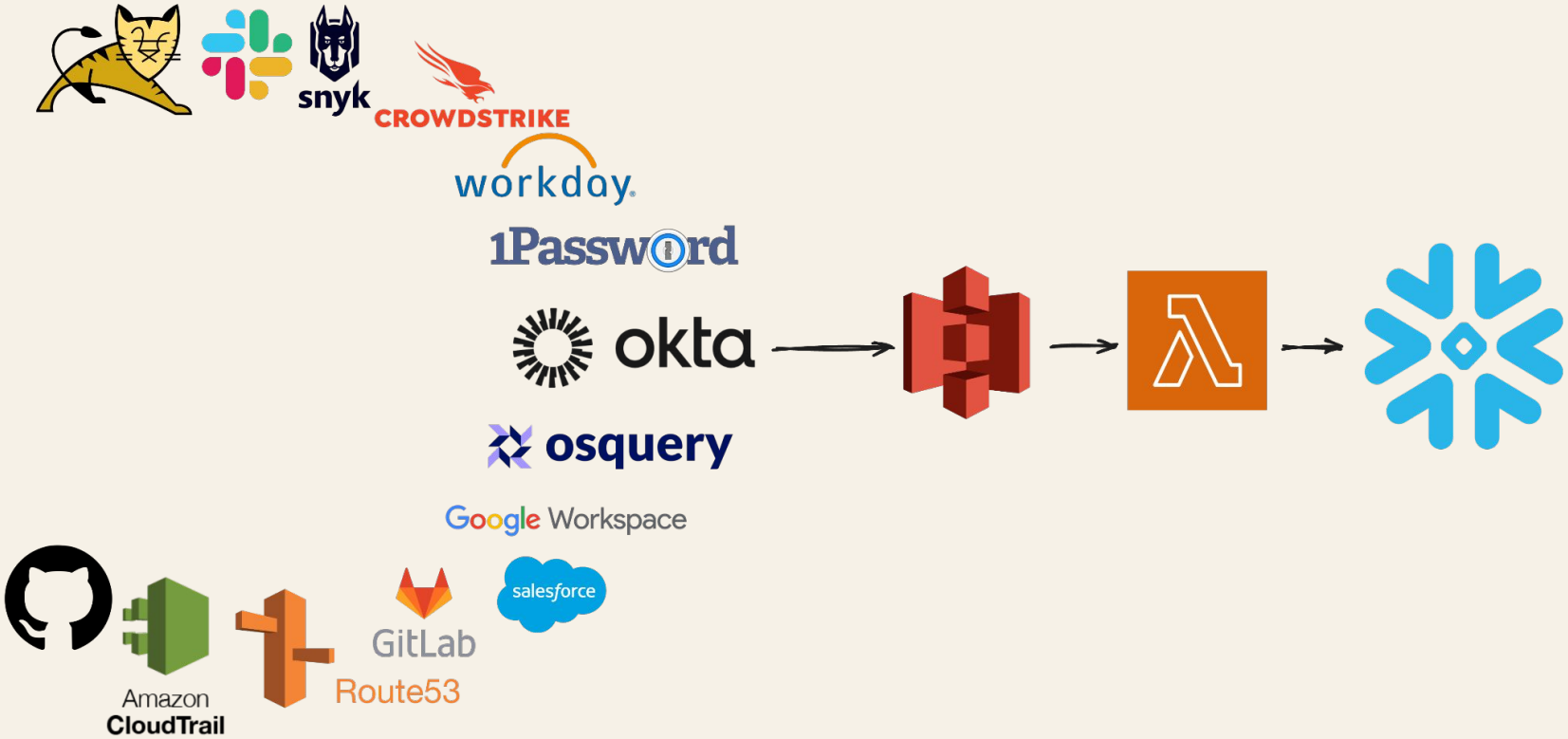
- Principal Eng, DCO
- Manager, Foundations
- Like building stuff
- Shiny things 😍😵

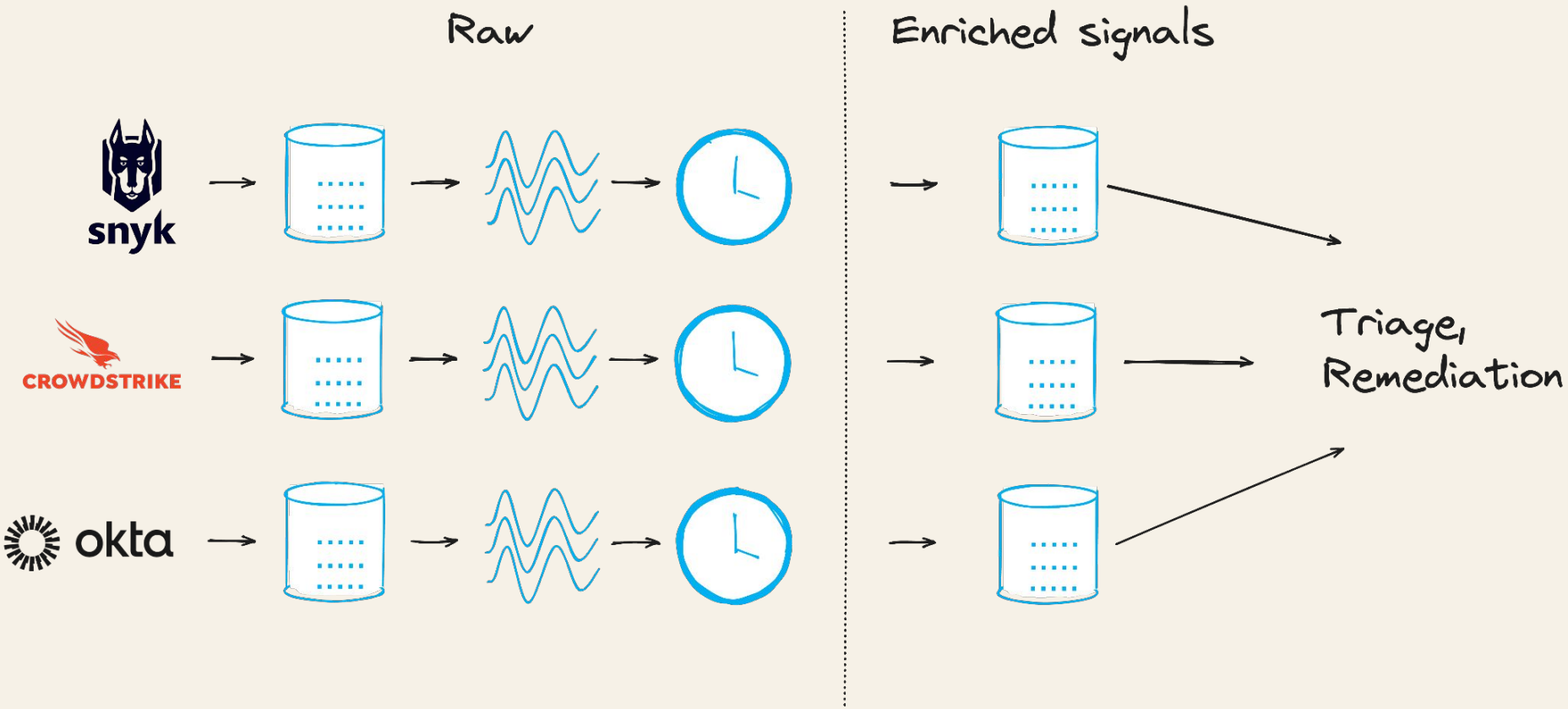
tl;dr

- Use case and goals
- Why embed OLAP?
- Tools and architecture
- Learnings
- Looking ahead

Defensive Cyber Operations @ Okta

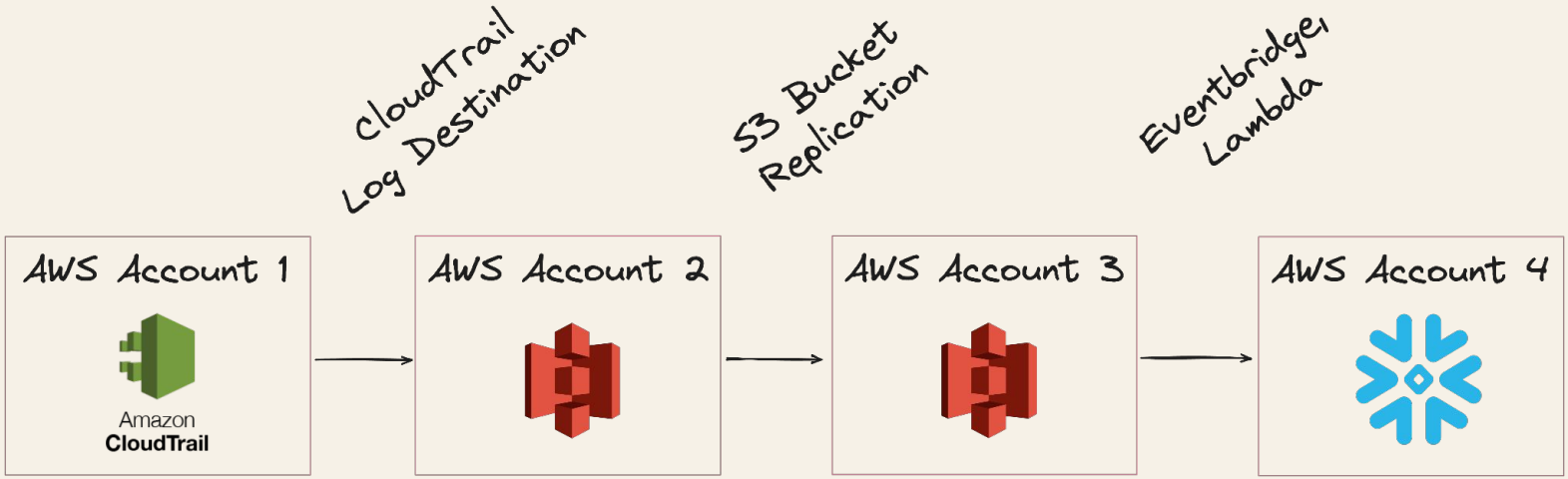
- Integrate diverse sources
- Normalize, enrich, model
- Monitor behavior, anomalies, tripwires
- Trigger workflows
- Be single entry point


snyk
CROWDSTRIKE
workday.
1Password
okta
osquery
Google Workspace
salesforce
GitLab
Amazon
CloudTrail
Route53

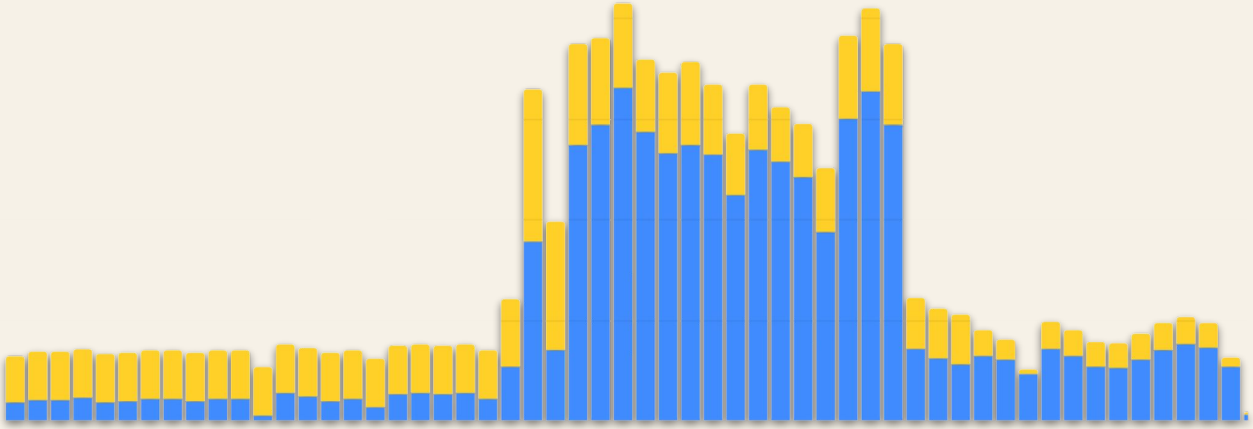

Raw
Enriched signals
snyk
CROWDSTRIKE
okta
Triage,
Remediation

What could possibly go wrong?

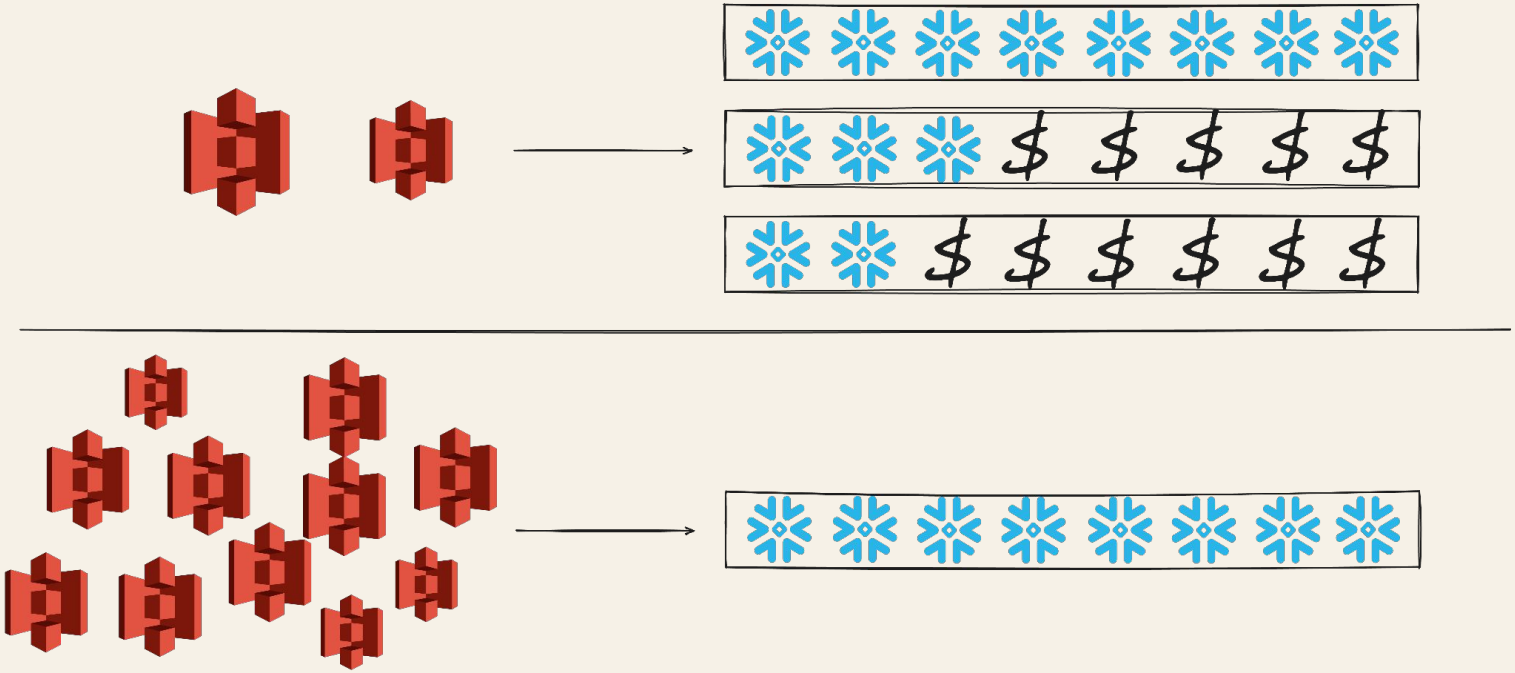
Complexity, Compliance



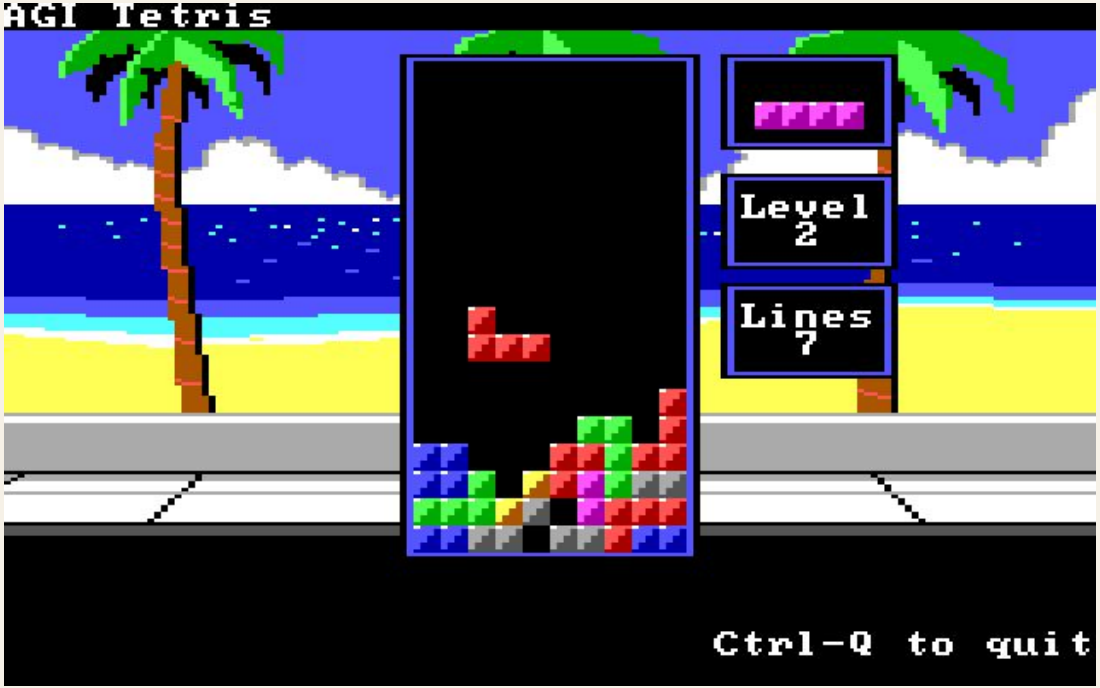
Spend



Pick One



S3 ~~Data Lake~~ Tetris Game



Actual Workloads

Small and Continuous

- Normalize file formats, shape
- Enrichment
- Generate metadata
- Harvest signals

Large and Periodic

- Historical Analytics
- Threat intelligence
- Model training
- Data sharing

Actual Workloads

Sm ontinuous

- No file formats, shape
-
- Generate metada
- gna

Large and Periodic

- Historical Analytics
- Threat intelligence
- Model training
- Data sharing

Tools, Architecture

Source → S3 NS

Source → SQL

S3 NS → Lambda

now() as processed_at

left join 's3://some-other-source.parquet'

unnest(mess)

qualify instance = 1

harvest signals

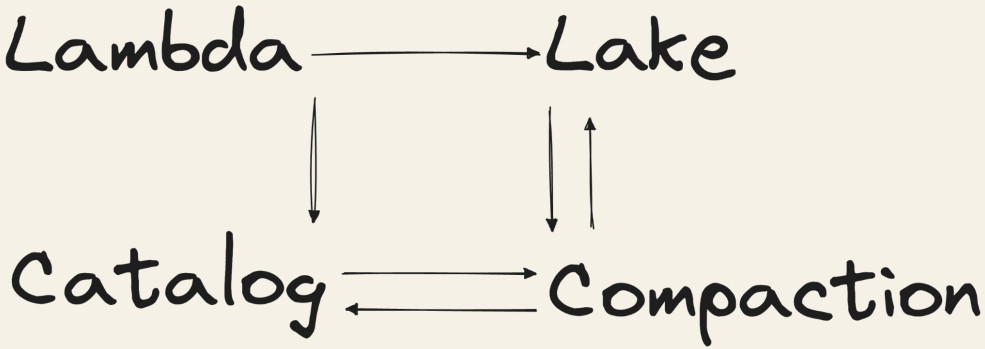
schematize

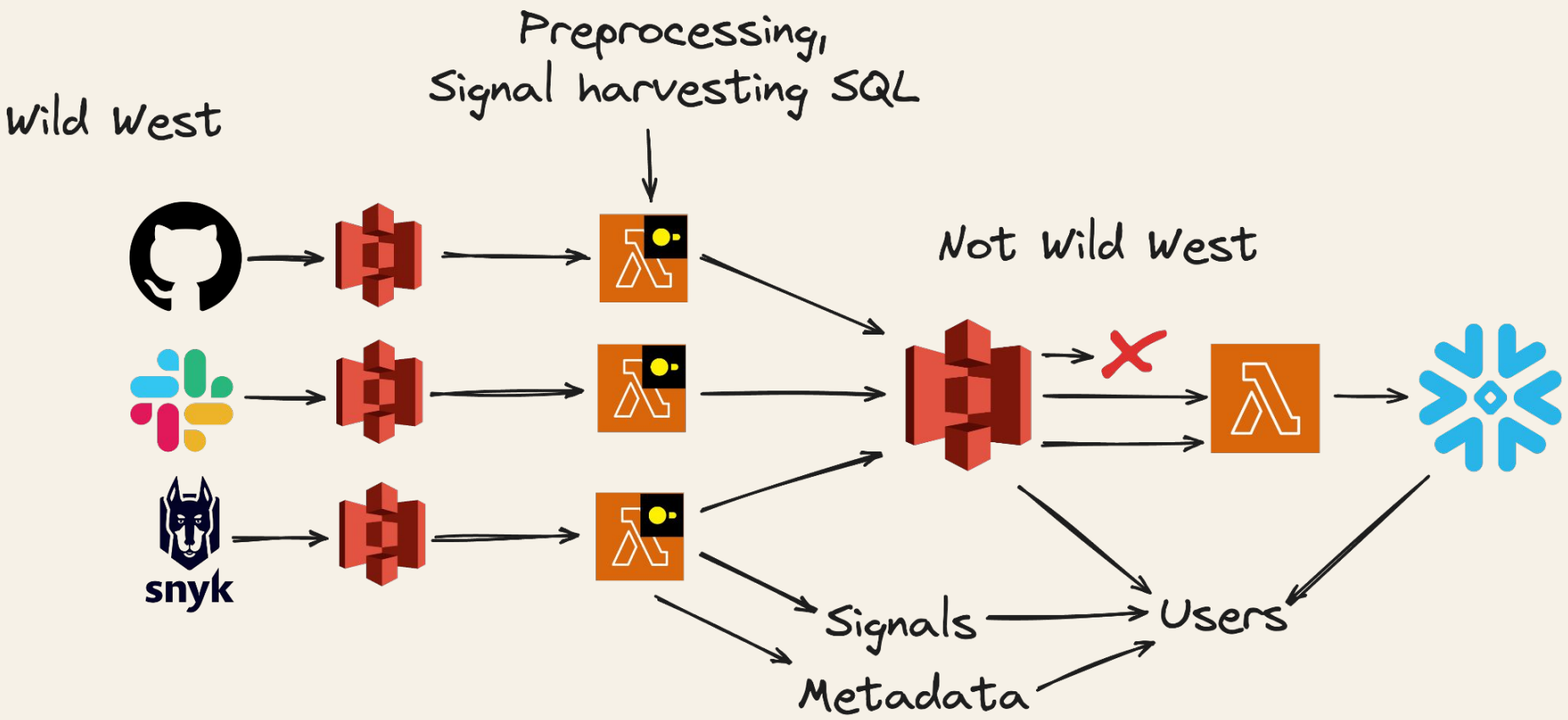
count(*) as rows

normalize

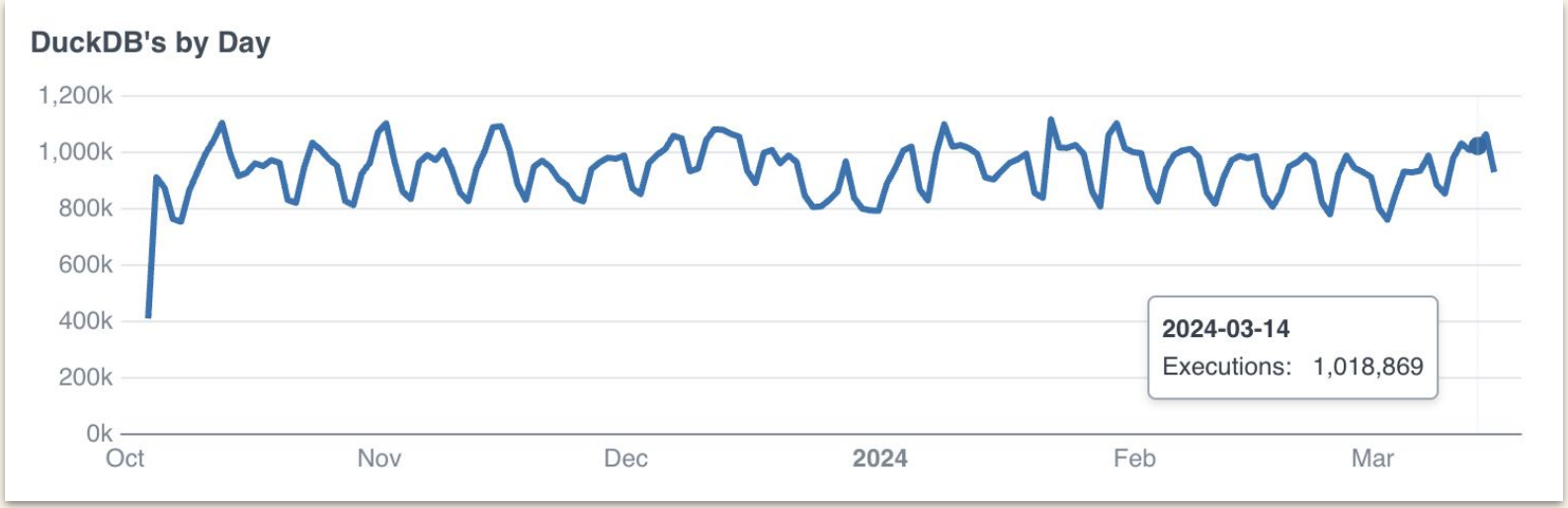
md5(payload) as fingerprint

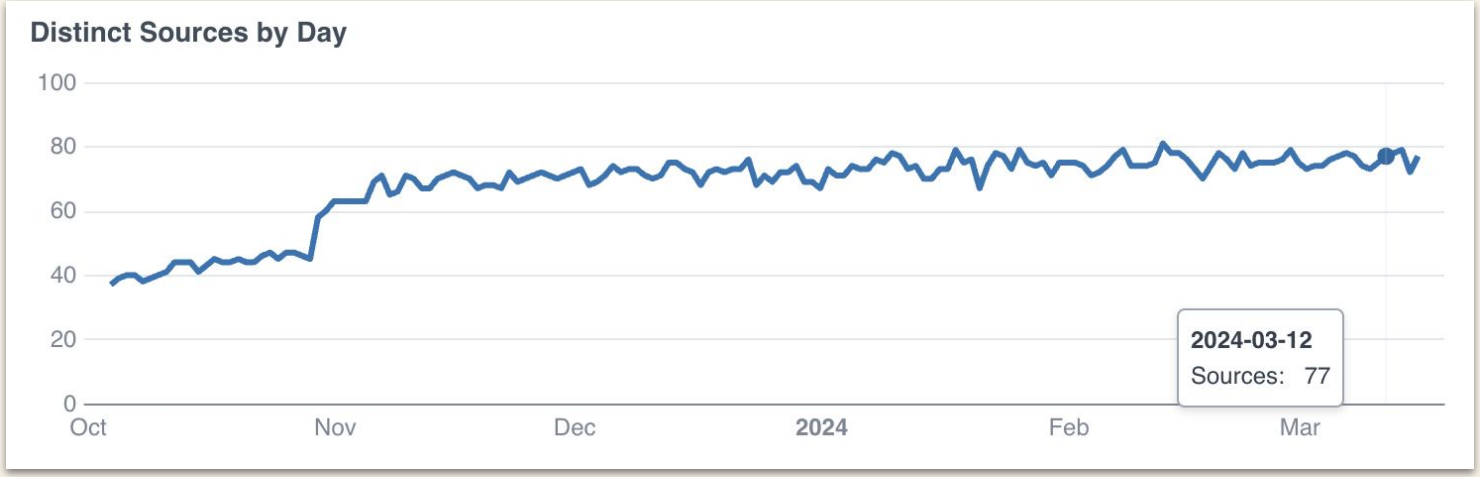
md5(string_agg(column_name)) as schema

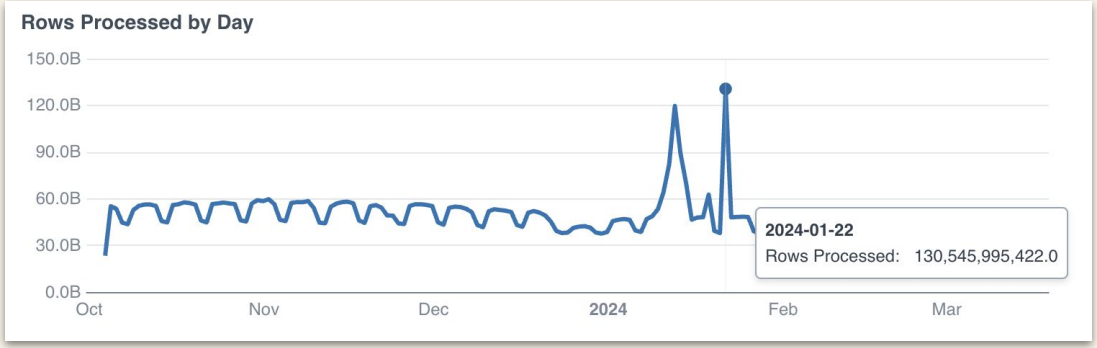

Lambda
Lake
Catalog
Compaction

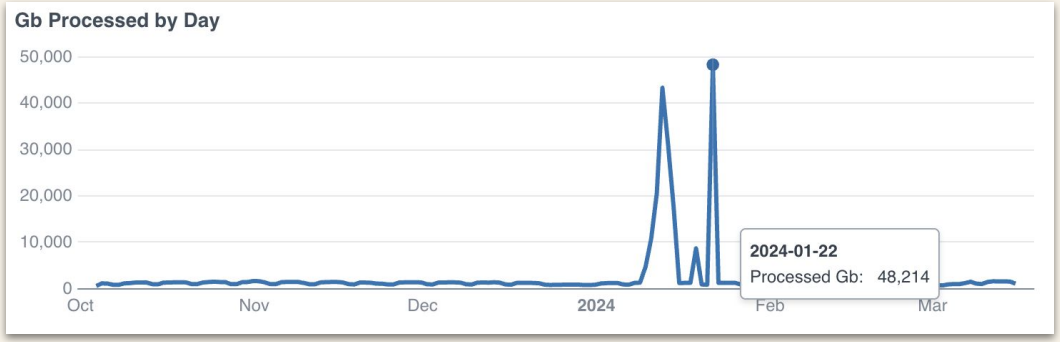

Wild West
Preprocessing,
Signal harvesting SQL
Not Wild West
snyk
Signals
Metadata
Users

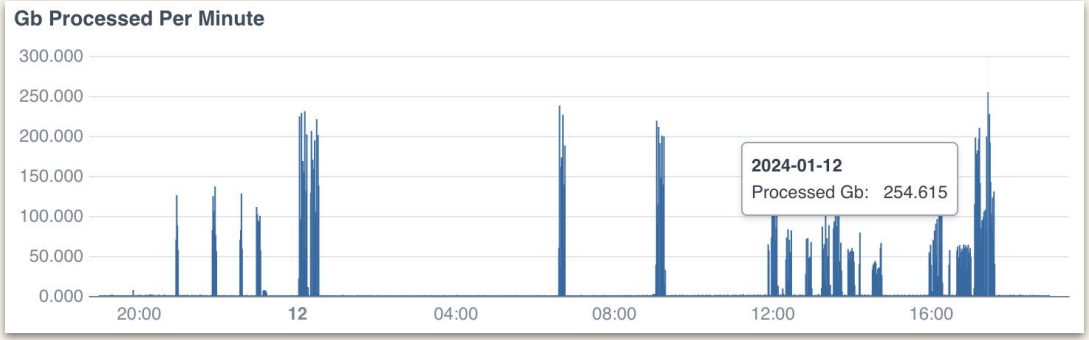
Stats, Learnings


DuckDB's by Day
1,200k
1,000k
800k
600k
400k
200k
0k
Oct
Nov
Dec
2024
Feb
Mar
2024-03-14
Executions: 1,018,869

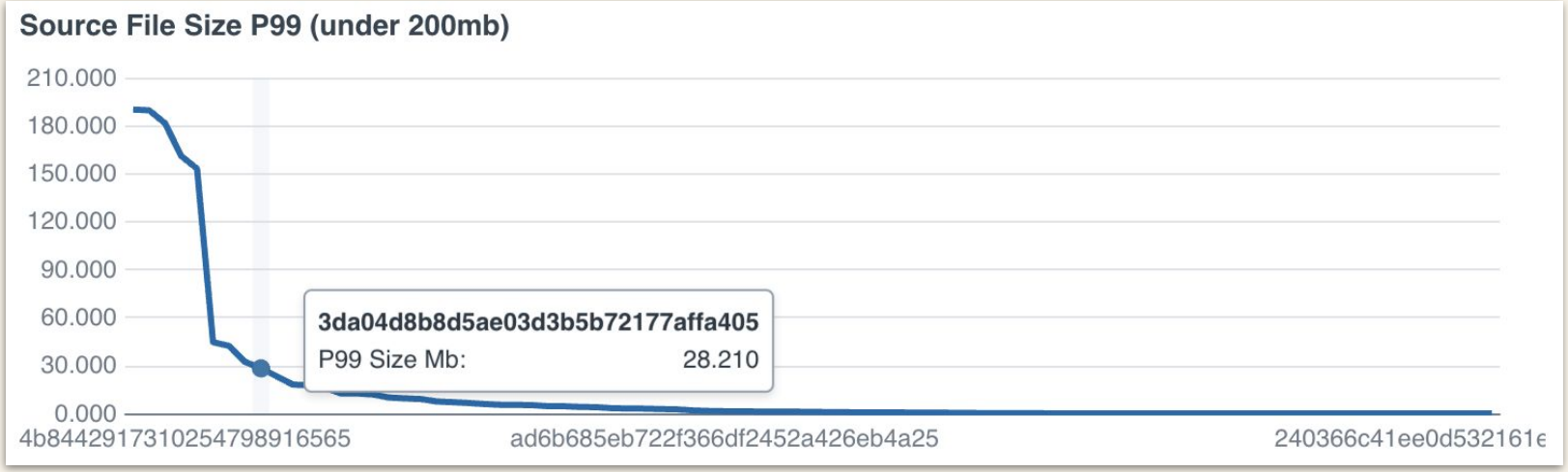

Distinct Sources by Day
100
80
60
40
20
0
Oct
Nov
Dec
2024
Feb
Mar
2024-03-12
Sources: 77

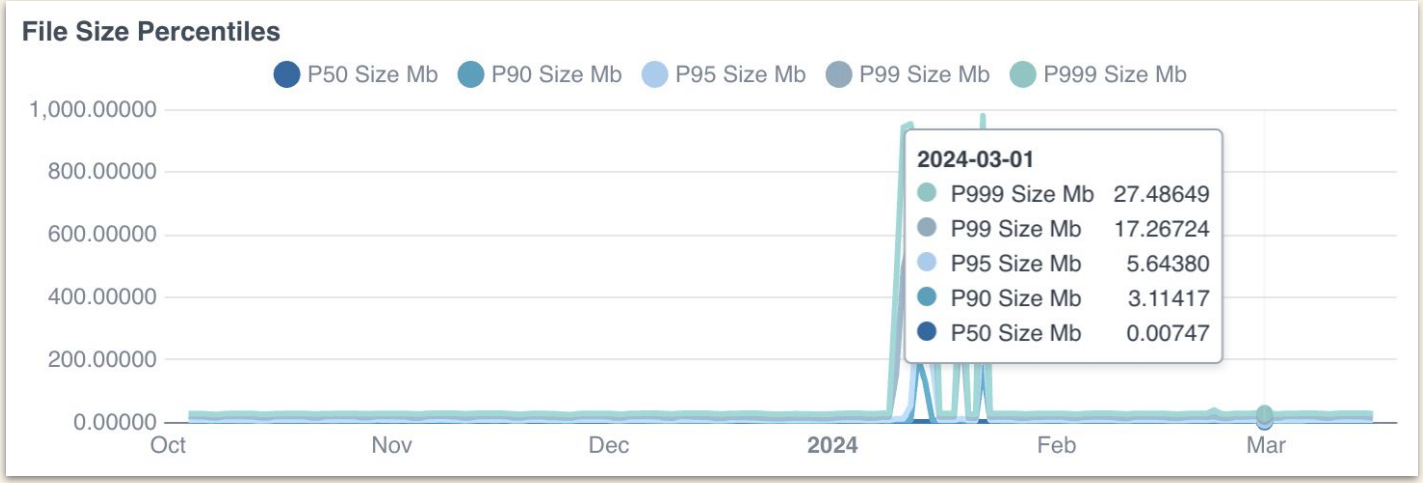

Rows Processed by Day
150.0B
120.0B
90.0B
60.0B
30.0B
0.0B
Oct
Nov
Dec
2024
Feb
Mar
2024-01-22
Rows Processed: 130,545,995,422.0

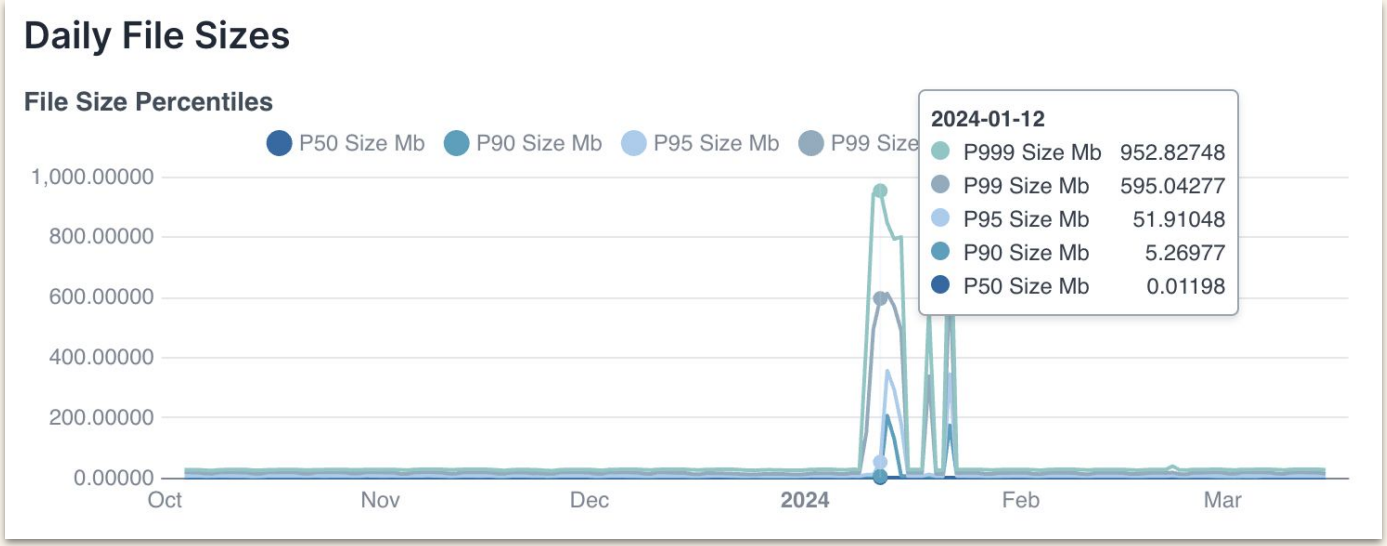

Gb Processed by Day
50,000
40,000
30,000
20,000
10,000
0
Oct
Nov
Dec
2024
Feb
Mar
2024-01-22
Processed Gb: 48,214

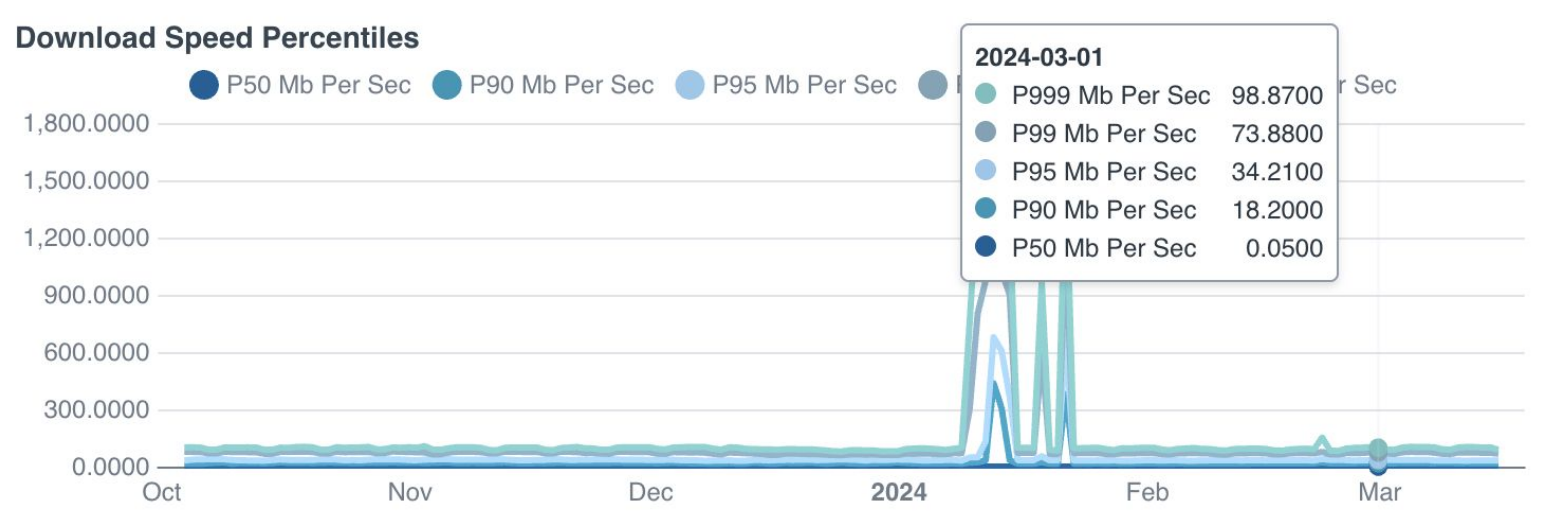

Gb Processed Per Minute
300.000
250.000
200.000
150.000
100.000
50.000
0.000
20:00
12
04:00
08:00
12:00
16:00
2024-01-12
Processed Gb: 254.615

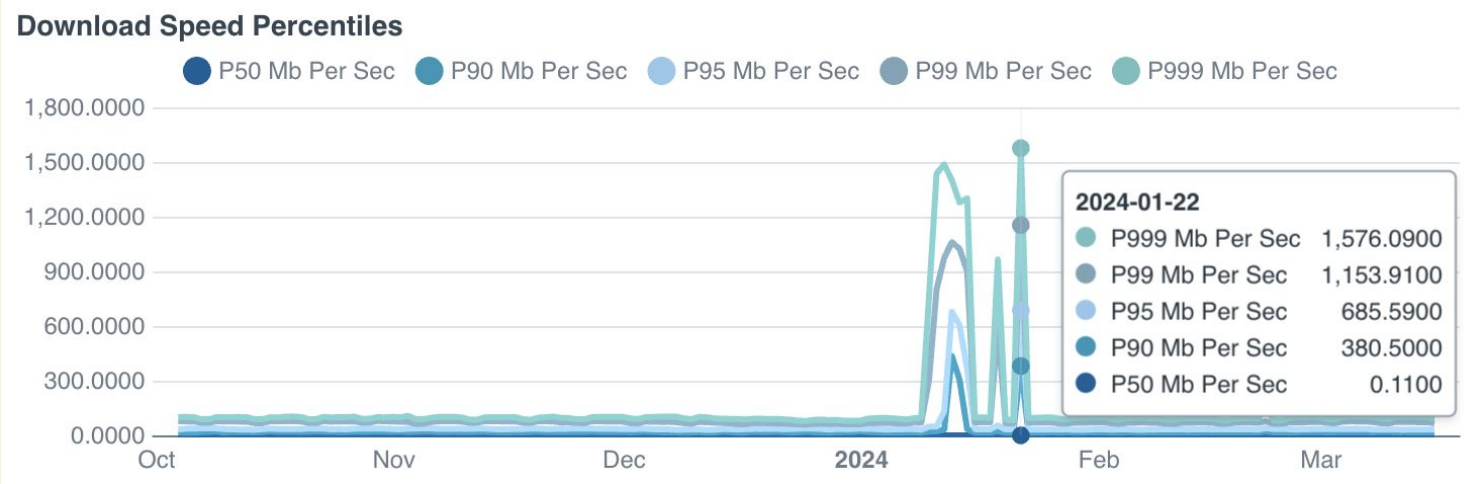

Source File Size P99
3,000.00
2,500.00
2,000.00
1,500.00
1,000.00
500.00
0.00
5fadbaa54b8442917310254798916565
P99 Size Mb: 190.02
84d45ccdc6fe2d8585341b334
7961d7471f9e867eed478cd93d27da75
240366c41ee0d532161e

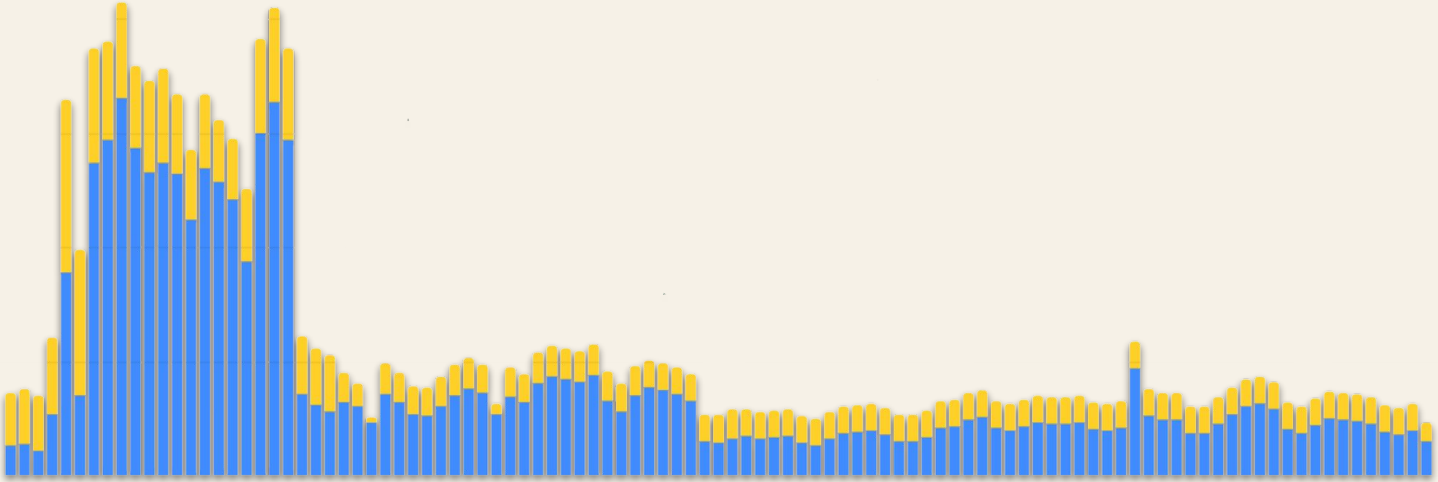

Source File Size P99 (under 200mb)
210.000
180.000
150.000
120.000
90.000
60.000
30.000
0.000
3da04d8b8d5ae03d3b5b72177affa405
P99 Size Mb: 28.210
4b8442917310254798916565
ad6b685eb722f366df2452a426eb4a25
240366c41ee0d532161e


File Size Percentiles
P50 Size Mb
P90 Size Mb
P95 Size Mb
P99 Size Mb
P999 Size Mb
1,000.00000
800.00000
600.00000
400.00000
200.00000
0.00000
Oct
Nov
Dec
2024
Feb
Mar
2024-03-01
P999 Size Mb 27.48649
P99 Size Mb 17.26724
P95 Size Mb 5.64380
P90 Size Mb 3.11417
P50 Size Mb 0.00747


Daily File Sizes
File Size Percentiles
P50 Size Mb
P90 Size Mb
P95 Size Mb
P99 Size
1,000.00000
800.00000
600.00000
400.00000
200.00000
0.00000
Oct
Nov
Dec
2024
Feb
Mar
2024-01-12
P999 Size Mb 952.82748
P99 Size Mb 595.04277
P95 Size Mb 51.91048
P90 Size Mb 5.26977
P50 Size Mb 0.01198


Download Speed Percentiles
P50 Mb Per Sec
P90 Mb Per Sec
P95 Mb Per Sec
1,800.0000
1,500.0000
1,200.0000
900.0000
600.0000
300.0000
0.0000
Oct
Nov
Dec
2024
Feb
Mar
2024-03-01
P999 Mb Per Sec 98.8700
P99 Mb Per Sec 73.8800
P95 Mb Per Sec 34.2100
P90 Mb Per Sec 18.2000
P50 Mb Per Sec 0.0500


Download Speed Percentiles
P50 Mb Per Sec
P90 Mb Per Sec
P95 Mb Per Sec
P99 Mb Per Sec
P999 Mb Per Sec
1,800.0000
1,500.0000
1,200.0000
900.0000
600.0000
300.0000
0.0000
Oct
Nov
Dec
2024
Feb
Mar
2024-01-22
P999 Mb Per Sec 1,576.0900
P99 Mb Per Sec 1,153.9100
P95 Mb Per Sec 685.5900
P90 Mb Per Sec 380.5000
P50 Mb Per Sec 0.1100



What would make
life better?

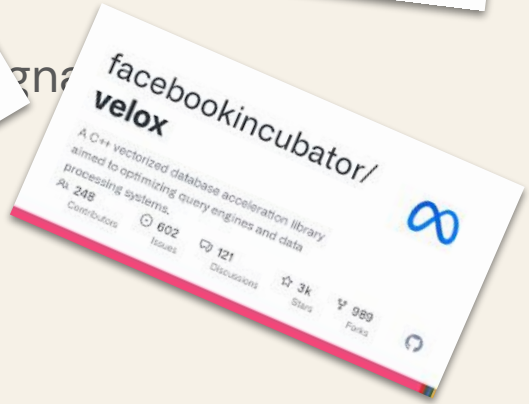
Actual Workloads

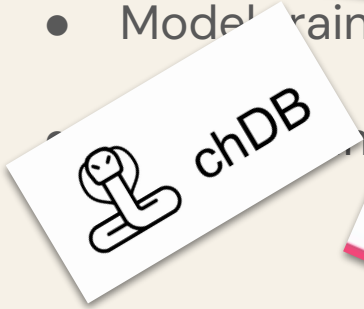
Sm ontinuous

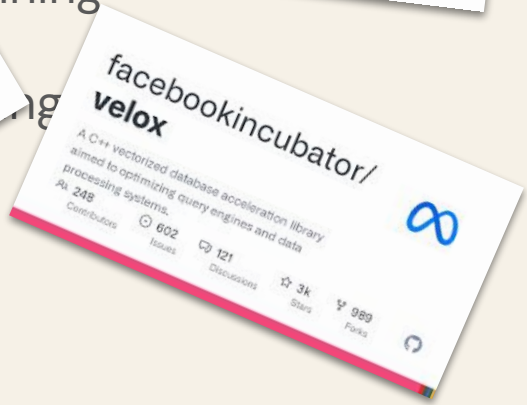
- No file formats, shape
-
- Generate metada
- gna

Larg riodic

- Ana s
-
- Model raining
- ng







Dynamic Resourcing

Memory Info

Your function is allocated CPU proportional to the memory configured.

128 MB

Set memory to between 128 MB and 10240 MB

Ephemeral storage Info

You can configure up to 10 GB of ephemeral storage (/tmp) for your function. View pricing

512 MB

Set ephemeral storage (/tmp) to between 512 MB and 10240 MB.

SnapStart Info

Reduce startup time by having Lambda cache a snapshot of your function after the function has initialized. To evaluate whether your function code is resilient to snapshot operations, review the SnapStart compatibility considerations.

None

Supported runtimes: Java 11, Java 17, Java 21.

Timeout

0 min 3 sec

File system

You can associate an existing Amazon Elastic File System (Amazon EFS) file system with your function. Visit the Amazon EFS console to create a new file system.

EFS file system

Choose an existing EFS file system to use with your Lambda function.

Local mount path

Only absolute paths are supported.

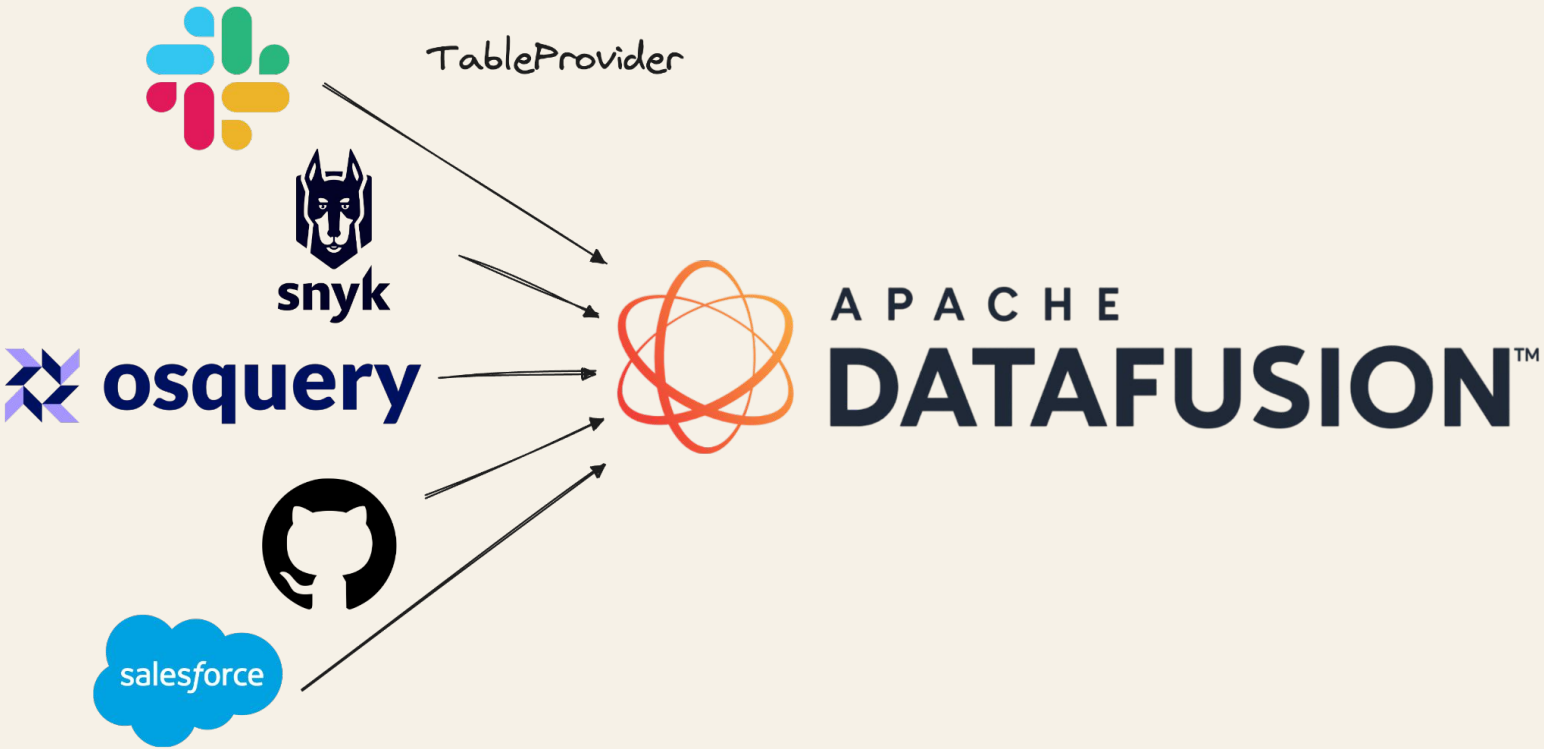
/mnt/

Concurrency

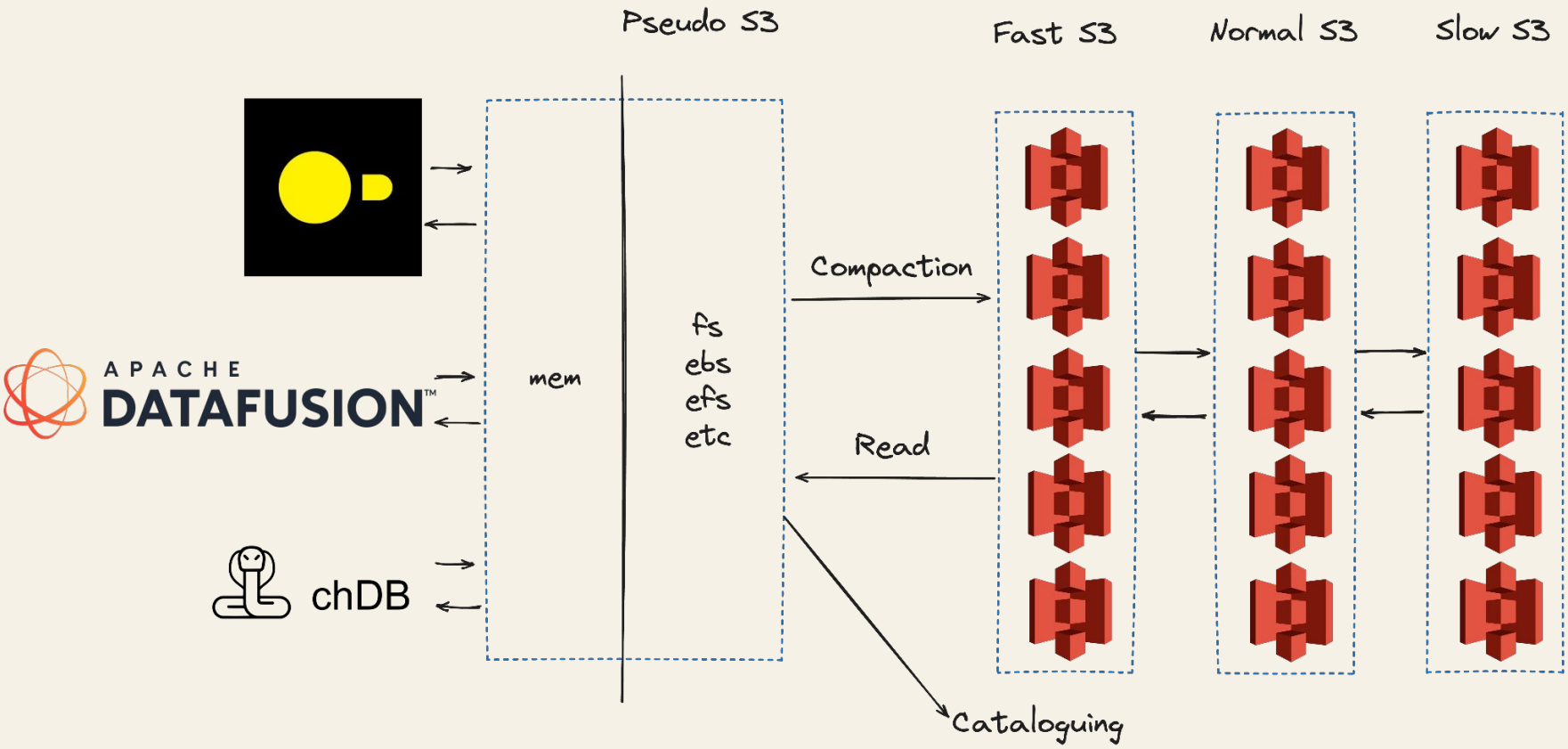
Unreserved account concurrency: **989**

- (●) Use unreserved account concurrency
- ( ) Reserve concurrency

Embedded Connector Ecosystem



Hybrid Storage



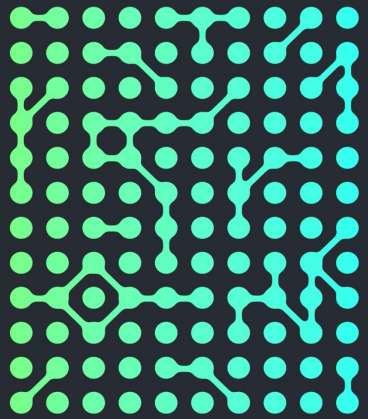
ACL Management

OpenFGA™

Relationship-based access control made fast, scalable, and easy to use.

OpenFGA is an open-source authorization solution that allows developers to build granular access control using an easy-to-read modeling language and friendly APIs.

Quick start | Join us on GitHub



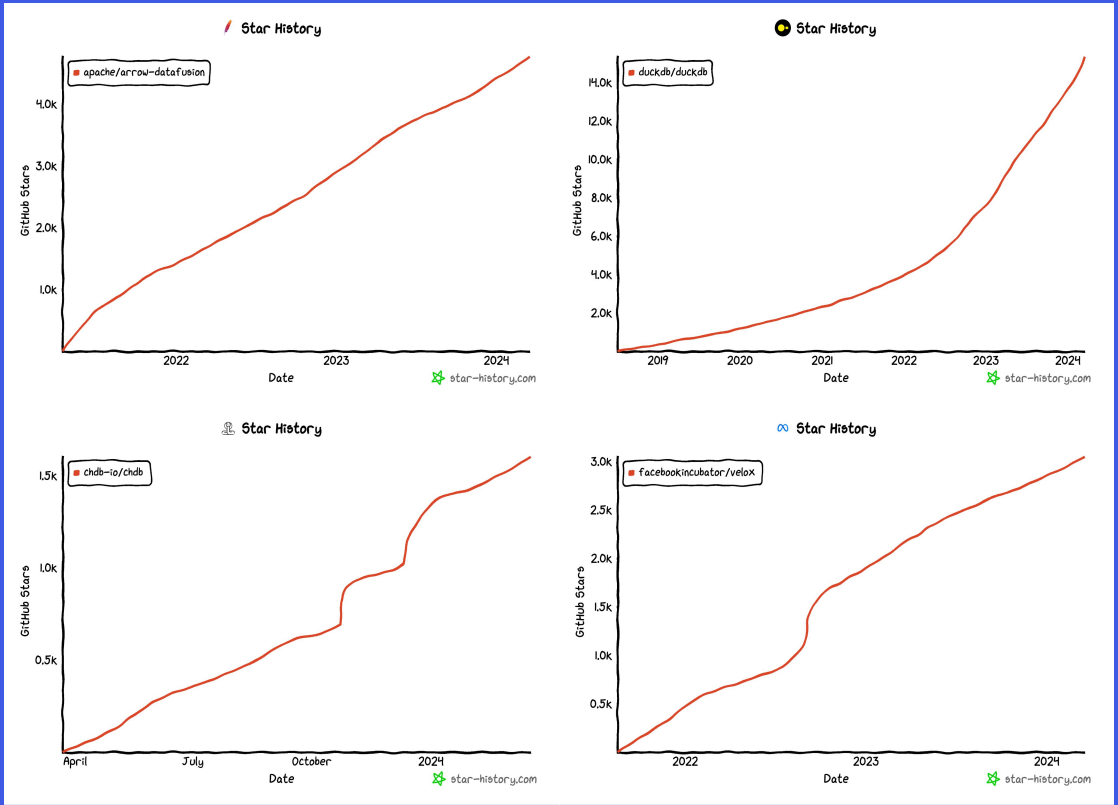
Publications > Zanzibar: Google's Consistent, Global Authorization System

Zanzibar: Google's Consistent, Global Authorization System

Ruoming Pang · Ramon Caceres · Mike Burrows · Zhifeng Chen · Pratik Dave · Nathan Germer · Alexander Golynski · Kevin Graney · Nina Kang · Lea Kissner · Jeffrey L. Korn · Abhishek Parmar · Christina D. Richards · Mengzhi Wang · 2019 USENIX Annual Technical Conference (USENIX ATC '19), Renton, WA

Download | Google Scholar | Copy Bibtex

Embedded OLAP at a Glance



Embedding OLAP at a Glance

Search the docs ...

LINKS

Github and Issue Tracker
crates.io
API Docs
Code of conduct

USER GUIDE

Introduction
Example Usage
Command line SQL console
DataFrame API
Expression API
SQL Reference
Configuration Settings
Frequently Asked Questions

LIBRARY USER GUIDE

Introduction
Using the SQL API
Working with `Expr` s
Using the DataFrame API
Building Logical Plans
Catalogs, Schemas, and Tables
Adding User Defined Functions: Scalar/Window/Aggregate/Table Functions
Custom Table Provider
Extending DataFusion's operators: custom LogicalPlan and Execution Plans

Known Users

Here are some active projects using DataFusion:

- Arroyo Distributed stream processing engine in Rust
- Ballista Distributed SQL Query Engine
- CnosDB Open Source Distributed Time Series Database
- Cube Store
- Dask SQL Distributed SQL query engine in Python
- Exon Analysis toolkit for life-science applications
- delta-rs Native Rust implementation of Delta Lake
- GreptimeDB Open Source & Cloud Native Distributed Time Series Database
- GlareDB Fast SQL database for querying and analyzing distributed data.
- HoraeDB Distributed Time-Series Database
- InfluxDB Time Series Database
- Kamu Planet-scale streaming data pipeline
- LakeSoul Open source LakeHouse framework with native IO in Rust.
- Lance Modern columnar data format for ML
- Parseable Log storage and observability platform
- ParadeDB PostgreSQL for Search & Analytics
- qv Quickly view your data
- bdt Boring Data Tool
- Restate Easily build resilient applications using distributed durable async/await
- ROAPI
- Seafowl CDN-friendly analytical database
- Synnada Streaming-first framework for data products
- VegaFusion Server-side acceleration for the Vega visualization grammar
- ZincObserve Distributed cloud native observability platform

Here are some less active projects that used DataFusion:

- Blaze Spark accelerator with DataFusion at its core
- Cloudfuse Buzz
- datafusion-tui Text UI for DataFusion
- Flock
- Tensorbase

On this page

Edit this page

Thank you