Open Data Foundations across Hudi, Iceberg, and Delta Lake

Data Council

Speaker Bio

Kyle Weller

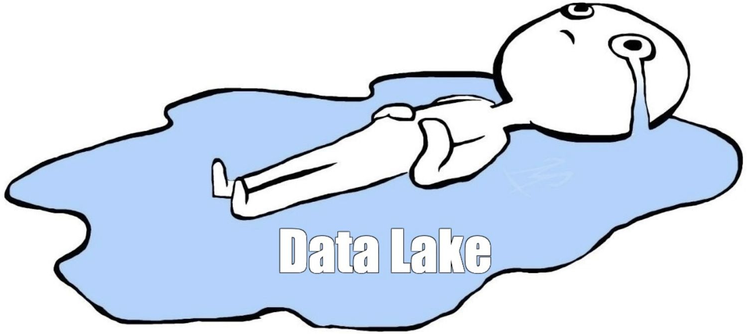
Head of Product @ ONEHOUSE

https://www.linkedin.com/in/lakehouse/

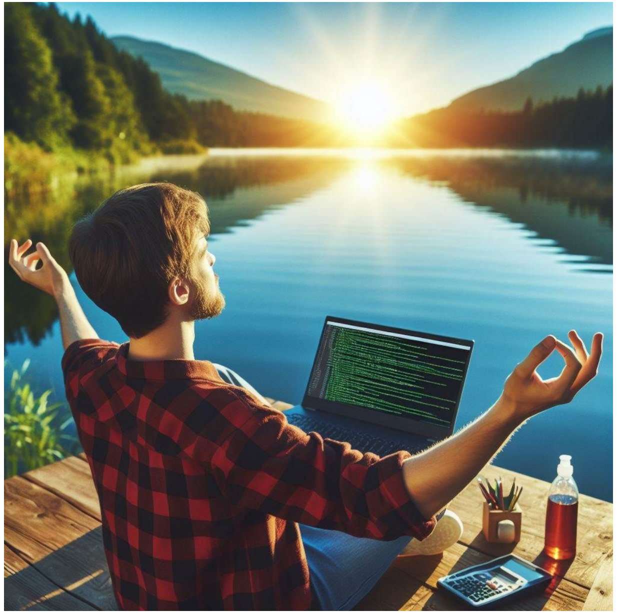
10+ years of building data platforms and data products

- Currently building Onehouse.ai
- Product lead for Azure Databricks 0 -> 9-fig ARR
- Built Azure Machine Learning services inside SQL Server
- Ran data and growth strategy for Cortana (MSFT AI assistant)
- Worked on PB scale data lake platform for Bing Search
- Designed v1 TB scale data lake for MSFT Office

Data Lakes…

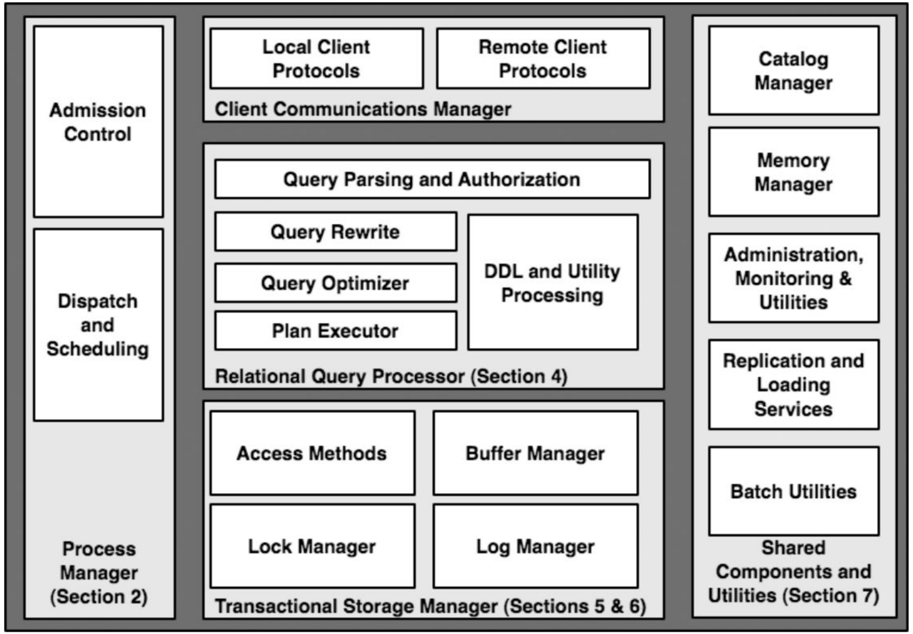


or



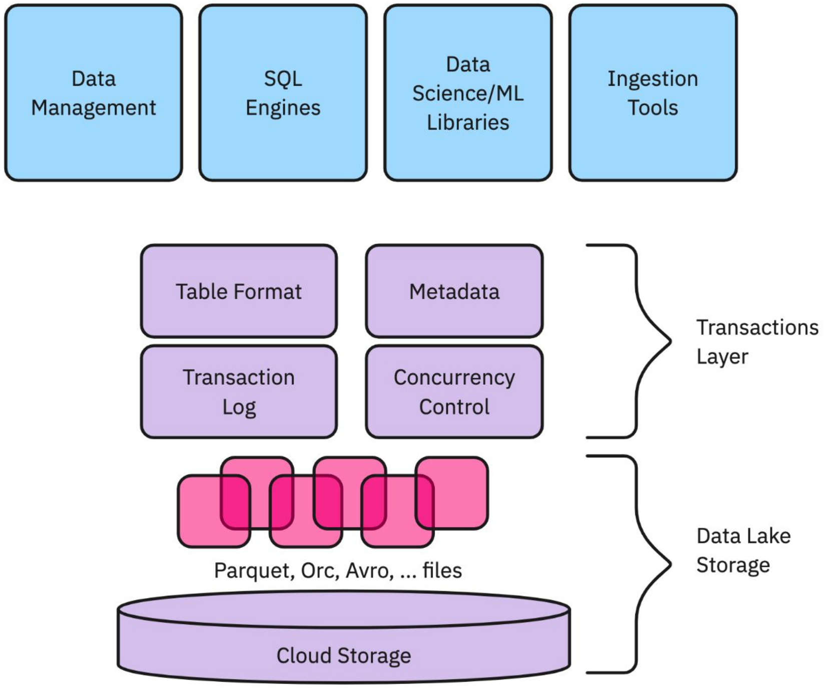
Architecture of a Database System

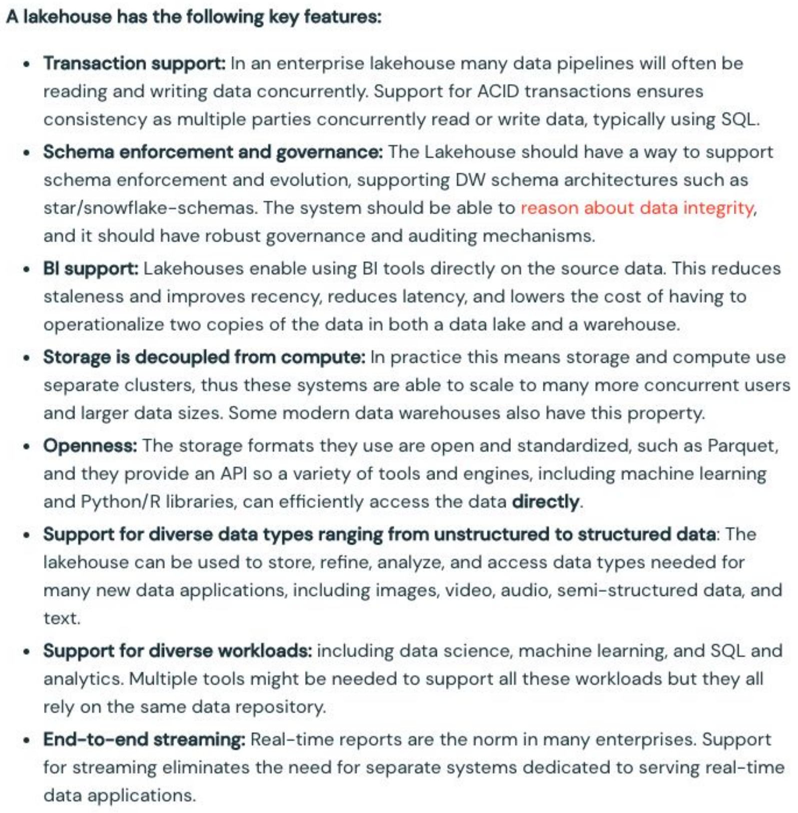
S3 Data Lake Storage


Admission Control
Dispatch and Scheduling
Process Manager (Section 2)
Local Client Protocols
Remote Client Protocols
Client Communications Manager
Query Parsing and Authorization
Query Rewrite
Query Optimizer
Plan Executor
DDL and Utility Processing
Relational Query Processor (Section 4)
Access Methods
Buffer Manager
Lock Manager
Log Manager
Transactional Storage Manager (Sections 5 & 6)
Catalog Manager
Memory Manager
Administration, Monitoring & Utilities
Replication and Loading Services
Batch Utilities
Shared Components and Utilities (Section 7)

https://dsf.berkeley.edu/papers/fntdb07-architecture.pdf

Data Lakehouse - Unbundling of the DBMS



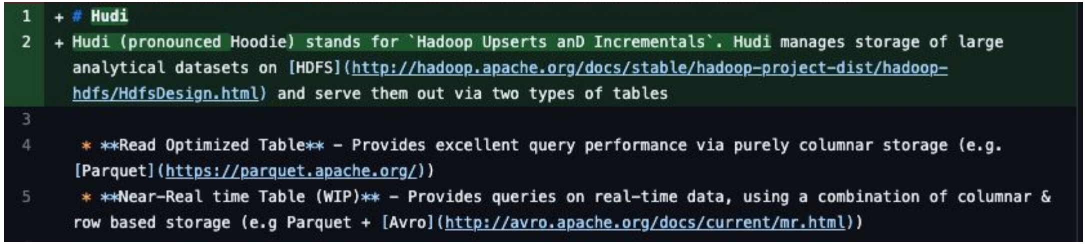


A lakehouse has the following key features:

- **Transaction support:** In an enterprise lakehouse many data pipelines will often be reading and writing data concurrently. Support for ACID transactions ensures consistency as multiple parties concurrently read or write data, typically using SQL.
- **Schema enforcement and governance:** The Lakehouse should have a way to support schema enforcement and evolution, supporting DW schema architectures such as star/snowflake-schemas. The system should be able to reason about data integrity, and it should have robust governance and auditing mechanisms.
- **BI support:** Lakehouses enable using BI tools directly on the source data. This reduces staleness and improves recency, reduces latency, and lowers the cost of having to operationalize two copies of the data in both a data lake and a warehouse.
- **Storage is decoupled from compute:** In practice this means storage and compute use separate clusters, thus these systems are able to scale to many more concurrent users and larger data sizes. Some modern data warehouses also have this property.
- **Openness:** The storage formats they use are open and standardized, such as Parquet, and they provide an API so a variety of tools and engines, including machine learning and Python/R libraries, can efficiently access the data **directly**.
- **Support for diverse data types ranging from unstructured to structured data**: The lakehouse can be used to store, refine, analyze, and access data types needed for many new data applications, including images, video, audio, semi-structured data, and text.
- **Support for diverse workloads:** including data science, machine learning, and SQL and analytics. Multiple tools might be needed to support all these workloads but they all rely on the same data repository.
- **End-to-end streaming:** Real-time reports are the norm in many enterprises. Support for streaming eliminates the need for separate systems dedicated to serving real-time data applications.

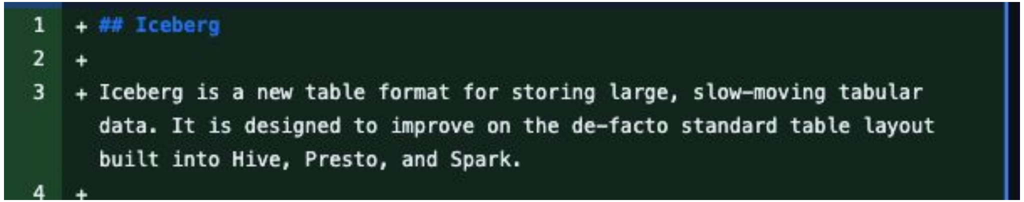
Origin Stories

2017 — open sourced Uber



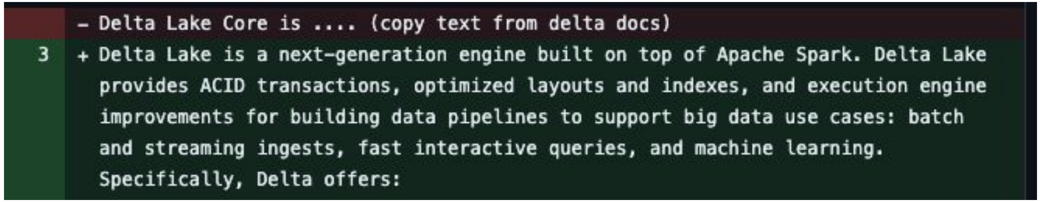
```
+ # Hudi
+ Hudi (pronounced Hoodie) stands for `Hadoop Upserts anD Incrementals`. Hudi manages storage of large
  analytical datasets on [HDFS](http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-
  hdfs/HdfsDesign.html) and serve them out via two types of tables

   * **Read Optimized Table** - Provides excellent query performance via purely columnar storage (e.g.
  [Parquet](https://parquet.apache.org/))
   * **Near-Real time Table (WIP)** - Provides queries on real-time data, using a combination of columnar &
  row based storage (e.g Parquet + [Avro](http://avro.apache.org/docs/current/mr.html))
```

2018 — open sourced N



```
+ ## Iceberg
+
+ Iceberg is a new table format for storing large, slow-moving tabular
  data. It is designed to improve on the de-facto standard table layout
  built into Hive, Presto, and Spark.
+
```

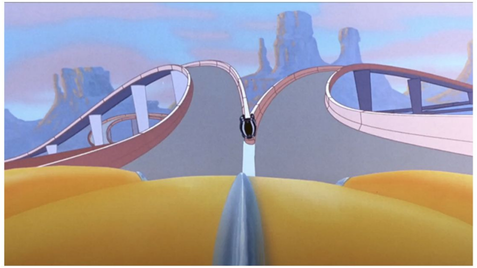
2019 — open sourced

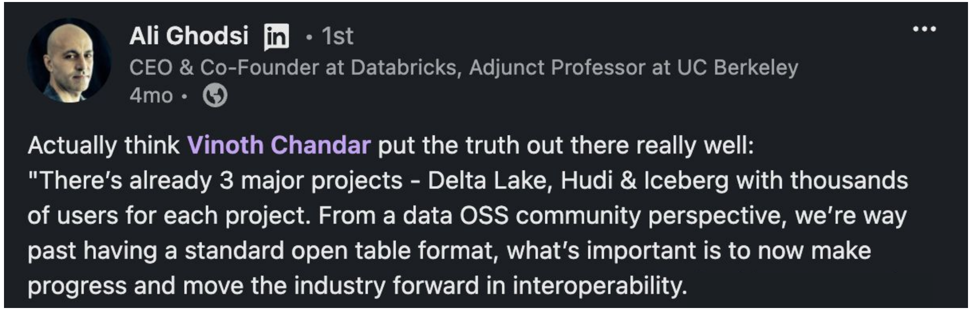


```
- Delta Lake Core is .... (copy text from delta docs)
+ Delta Lake is a next-generation engine built on top of Apache Spark. Delta Lake
  provides ACID transactions, optimized layouts and indexes, and execution engine
  improvements for building data pipelines to support big data use cases: batch
  and streaming ingests, fast interactive queries, and machine learning.
  Specifically, Delta offers:
```

🌶️ My Hot Take - They are divergent!

- Technical vision and goals are divergent
- The community needs are specialized
- All three projects are on fast growth trajectories
- New table formats are gaining traction: Apache Paimon, YOHB?



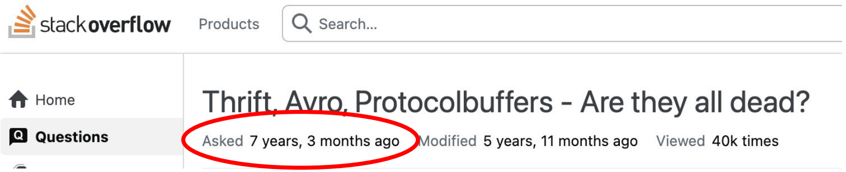


Ali Ghodsi in • 1st

CEO & Co-Founder at Databricks, Adjunct Professor at UC Berkeley

4mo •

Actually think Vinoth Chandar put the truth out there really well:
"There's already 3 major projects - Delta Lake, Hudi & Iceberg with thousands of users for each project. From a data OSS community perspective, we're way past having a standard open table format, what's important is to now make progress and move the industry forward in interoperability.

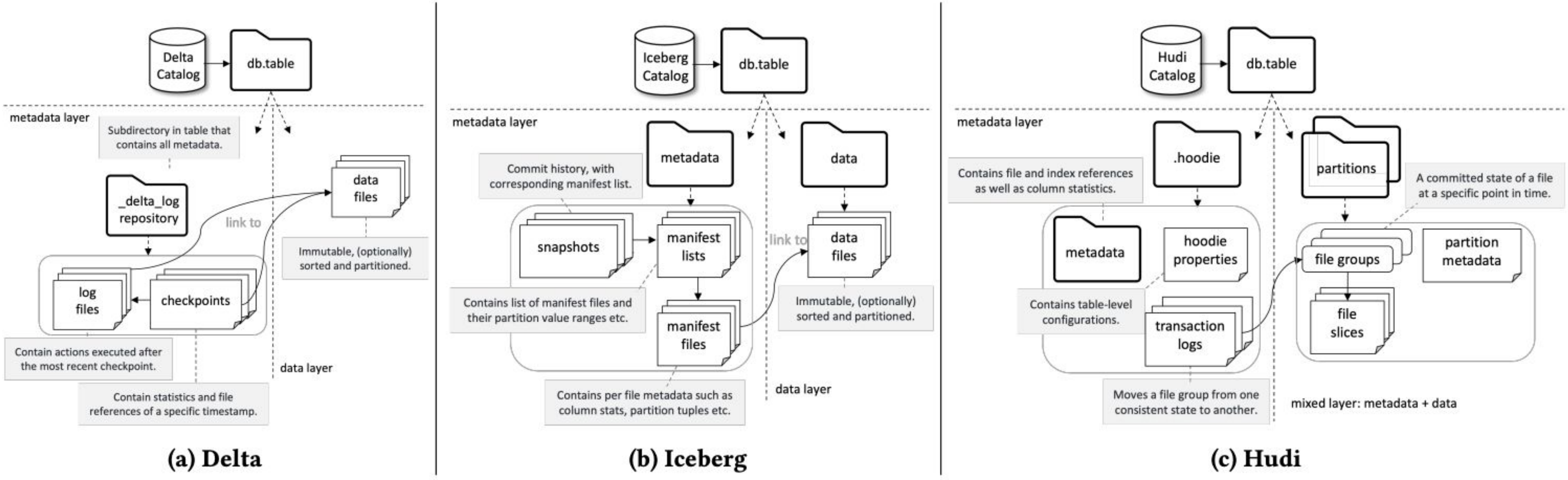


Thrift, Avro, Protocolbuffers - Are they all dead?

Asked 7 years, 3 months ago Modified 5 years, 11 months ago Viewed 40k times

Technical Fundamentals

- Metadata abstractions on files in cloud object storage
- Tables with SQL semantics and schema evolution
- ACID transactions
- Updates and deletes (merge/upsert)
- Data layout optimizations for performance tuning


Delta Catalog
db.table
metadata layer
Subdirectory in table that contains all metadata.
_delta_log repository
link to
data files
Immutable, (optionally) sorted and partitioned.
log files
checkpoints
Contain actions executed after the most recent checkpoint.
data layer
Contain statistics and file references of a specific timestamp.
Iceberg Catalog
db.table
metadata layer
metadata
data
Commit history, with corresponding manifest list.
snapshots
manifest lists
link to
data files
Contains list of manifest files and their partition value ranges etc.
Immutable, (optionally) sorted and partitioned.
manifest files
data layer
Contains per file metadata such as column stats, partition tuples etc.
Hudi Catalog
db.table
metadata layer
.hoodie
partitions
Contains file and index references as well as column statistics.
A committed state of a file at a specific point in time.
metadata
hoodie properties
file groups
partition metadata
Contains table-level configurations.
file slices
transaction logs
Moves a file group from one consistent state to another.
mixed layer: metadata + data

(a) Delta

(b) Iceberg

(c) Hudi

- How it looks on cloud storage

- Fundamentals of table formats Hudi, Delta, Iceberg are not that different
- Each has a special metadata layer on top of parquet files

Apache hudi

```
s3_bucket/my_table/
 |- .hoodie/
 |    |- hoodie.properties
 |    |- metadata/
 |- file_1.parquet
 |- file_2.parquet
 |- file_N.parquet
```

DELTA LAKE

```
s3_bucket/my_table/
 |- _delta_log/
 |    |- 000000.json
 |- file_1.parquet
 |- file_2.parquet
 |- file_N.parquet
```

ICEBERG

```
s3_bucket/my_table/
 |- metadata/
 |    |- v1.metadata.json
 |    |- snap-9fa1-2-16c3.avro
 |    |- 0d9a-98fa-77.avro
 |- file_1.parquet
 |- file_2.parquet
 |- file_N.parquet
```

Amazon S3 Google Cloud Storage Azure Data Lake Storage HDFS

- Which Format Should I Choose?

Choose Apache hudi if:

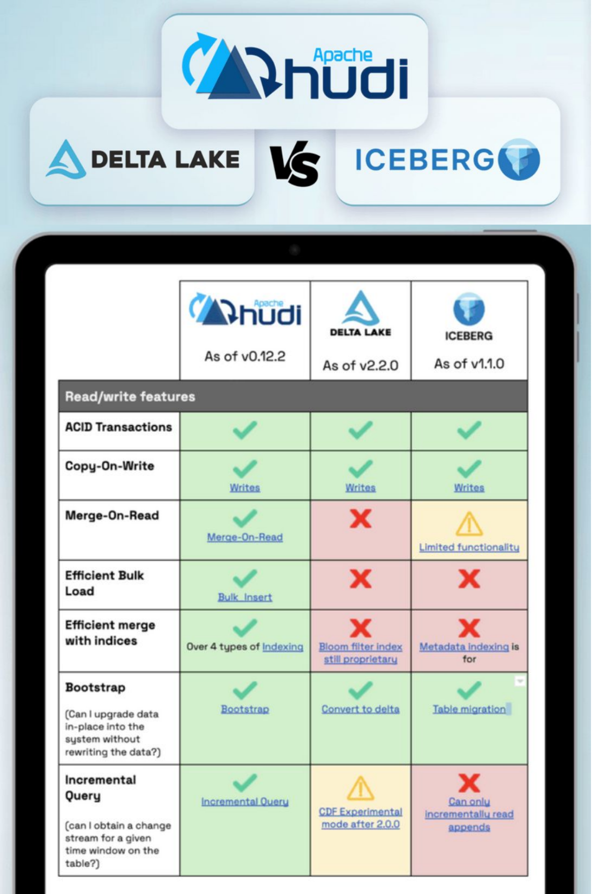
1. Mutable data - GDPR Deletes, Updates
2. CDC workloads
3. Low latency requirements
4. Large ETL pipelines - perf/cost w/ incremental ETL

Choose DELTA LAKE if:

1. Best Databricks experience
2. Needs fastest premium Spark with Photon
3. Wants an “easy-to-get-started” table format

Choose ICEBERG if:

1. Trino or Athena writes
2. Snowflake writes
3. Not sensitive to performance
4. Partition evolution


Apache hudi
DELTA LAKE
vs
ICEBERG

	Apache hudi As of v0.12.2	DELTA LAKE As of v2.2.0	ICEBERG As of v1.1.0
Read/write features			
ACID Transactions	✓	✓	✓
Copy-On-Write	✓ Writes	✓ Writes	✓ Writes
Merge-On-Read	✓ Merge-On-Read	✗	⚠ Limited functionality
Efficient Bulk Load	✓ Bulk Insert	✗	✗
Efficient merge with indices	✓ Over 4 types of Indexing	✗ Bloom filter index still proprietary	✗ Metadata indexing is for
Bootstrap (Can I upgrade data in-place into the system without rewriting the data?)	✓ Bootstrap	✓ Convert to delta	✓ Table migration
Incremental Query (can I obtain a change stream for a given time window on the table?)	✓ Incremental Query	⚠ CDF Experimental mode after 2.0.0	✗ Can only incrementally read appends

- Which Format Should I Choose?

Choos hudi if:

1. Mutable data - GDPR Deletes, Up s
2. DC workloads
3. L latency requirements
4. Larg TL pipelines - perf/cost w/ increme al ETL

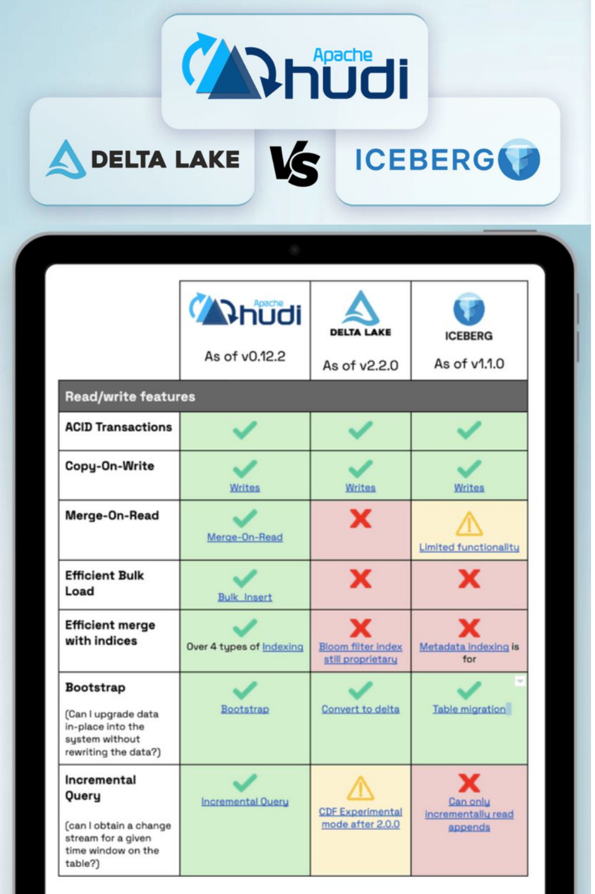
Choose DELTA E if:

2. Needs fastest pre Spark with Photon
3. Wants an “easy-to-ge rted” table format

What if you could work across all 3?

Choose ICEBERG if:

Trino or Athena writes
Snowflake writes
3. ensitive to performance
4. Parti


Apache hudi
DELTA LAKE
VS
ICEBERG

	Apache hudi As of v0.12.2	DELTA LAKE As of v2.2.0	ICEBERG As of v1.1.0
Read/write features			
ACID Transactions	✓	✓	✓
Copy-On-Write	✓ Writes	✓ Writes	✓ Writes
Merge-On-Read	✓ Merge-On-Read	✗	⚠ Limited functionality
Efficient Bulk Load	✓ Bulk Insert	✗	✗
Efficient merge with indices	✓ Over 4 types of Indexing	✗ Bloom filter index still proprietary	✗ Metadata indexing is for
Bootstrap (Can I upgrade data in-place into the system without rewriting the data?)	✓ Bootstrap	✓ Convert to delta	✓ Table migration
Incremental Query (can I obtain a change stream for a given time window on the table?)	✓ Incremental Query	⚠ CDF Experimental mode after 2.0.0	✗ Can only incrementally read appends

- Example benefits of mix-and-match

Writing

Choose Apache hudi writing w/ EMR (Spark)

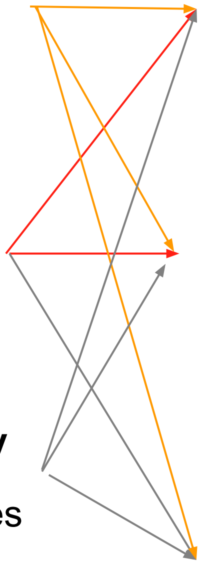
1. Fastest writes for mutable workloads
2. Most flexible tuning parameters for ingestion

Choose DELTA LAKE writing w/ Fabric:

1. Easy-to-get-started out of the box
2. Makes data available to the entire Azure portfolio

Choose ICEBERG writing w/ BigQuery

1. Only table format supported for writes
2. Partition evolution



Reading

Choose DELTA LAKE reading w/ Databricks

1. Get fastest queries with Photon acceleration
2. Great experience for Data Science

Choose ICEBERG reading w/ Snowflake

1. Only supported table format in Snowflake
2. Decouple data storage using external tables

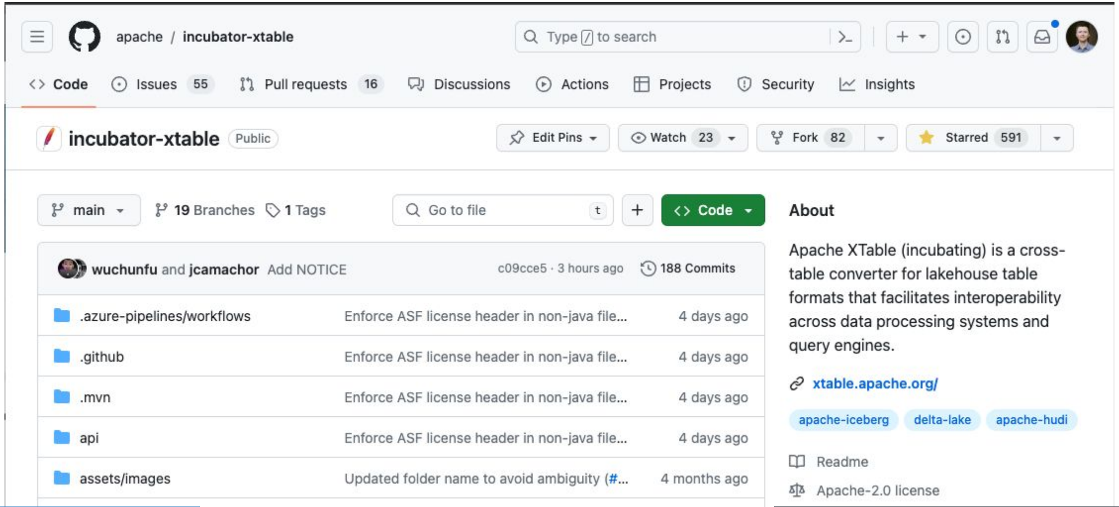
Choose Apache hudi reading w/ DataProc (Spark)

1. Fast record level indexes for point queries
2. Powerful secondary indexing capabilities for Spark

Introducing: Apache XTable™ Incubating

⭐Celebrate by adding a little star ⭐

https://github.com/apache/incubator-xtable



apache / incubator-xtable

Type / to search

Code | Issues 55 | Pull requests 16 | Discussions | Actions | Projects | Security | Insights

incubator-xtable Public

Edit Pins | Watch 23 | Fork 82 | Starred 591

main | 19 Branches | 1 Tags | Go to file | t | + | Code

wuchunfu and jcamachor Add NOTICE c09cce5 · 3 hours ago 188 Commits

.azure-pipelines/workflows	Enforce ASF license header in non-java file...	4 days ago
.github	Enforce ASF license header in non-java file...	4 days ago
.mvn	Enforce ASF license header in non-java file...	4 days ago
api	Enforce ASF license header in non-java file...	4 days ago
assets/images	Updated folder name to avoid ambiguity (#...	4 months ago
	Enforce ASF license header in non-java file...	4 days ago

About

Apache XTable (incubating) is a cross-table converter for lakehouse table formats that facilitates interoperability across data processing systems and query engines.

xtable.apache.org/

apache-iceberg | delta-lake | apache-hudi

Readme

Apache-2.0 license

>600 GH Stars ⭐

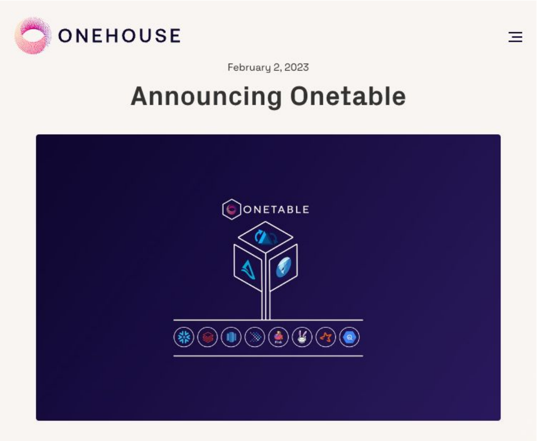
>80 Forks

OneTable @OnetableOSS

Today we made it on the TOP PAGE of Github Repos Trending worldwide: github.com/trending 🚀

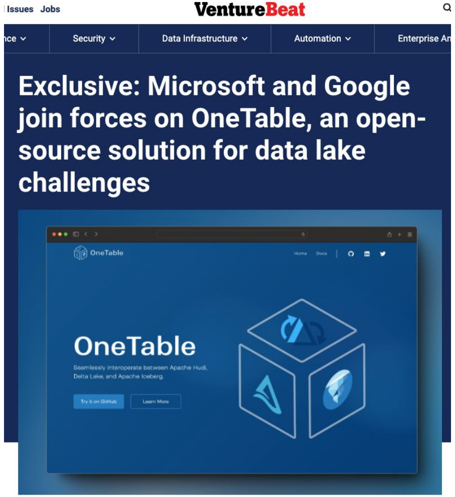
#OneTable #apachehudi #apacheiceberg #deltalake

Apache XTable™ - Timeline


ONEHOUSE

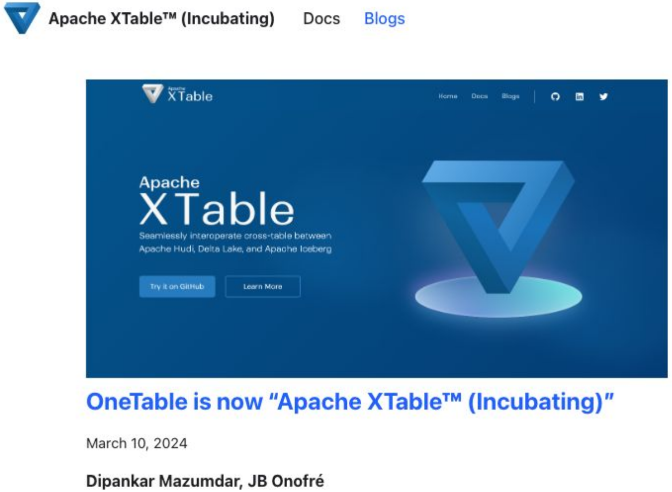
February 2, 2023

Announcing Onetable



Exclusive: Microsoft and Google join forces on OneTable, an open-source solution for data lake challenges

OneTable
Home
Docs
OneTable
Seamlessly interoperate between Apache Hudi, Delta Lake, and Apache Iceberg.
Try it on GitHub
Learn More


Apache XTable™ (Incubating)
Docs
Blogs
XTable
Home
Docs
Blogs
Apache
XTable
Seamlessly interoperate cross-table between Apache Hudi, Delta Lake, and Apache Iceberg
Try it on GitHub
Learn More

OneTable is now "Apache XTable™ (Incubating)"

March 10, 2024

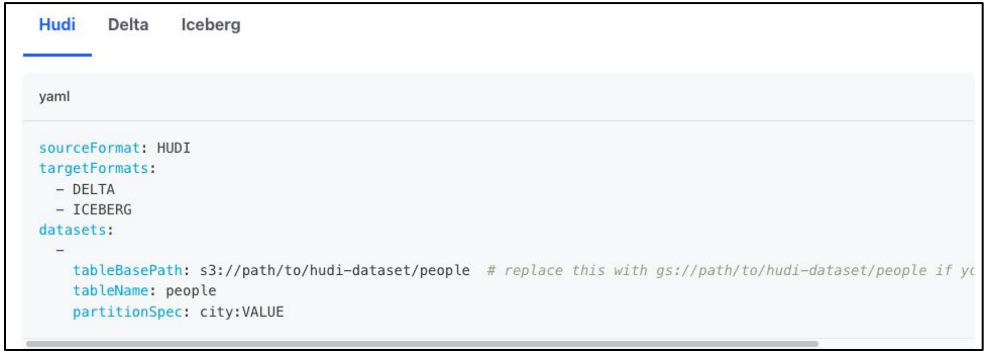
Dipankar Mazumdar, JB Onofré

Feb 2023	Nov 2023	Mar 2024
Onehouse announces OneTable	OSS Co-Launch with Microsoft, Google, Onehouse	Donation to ASF and incubation as Apache XTable
ONEHOUSE	Microsoft Google ONEHOUSE	Microsoft CLOUDERA Walmart ONEHOUSE dremio Adobe Google

Apache XTable™ - How it Works

1: Choose your "source" format
2: Choose your "target" format(s)
3: XTable translates the metadata layers

Read your table as any of the formats



yaml

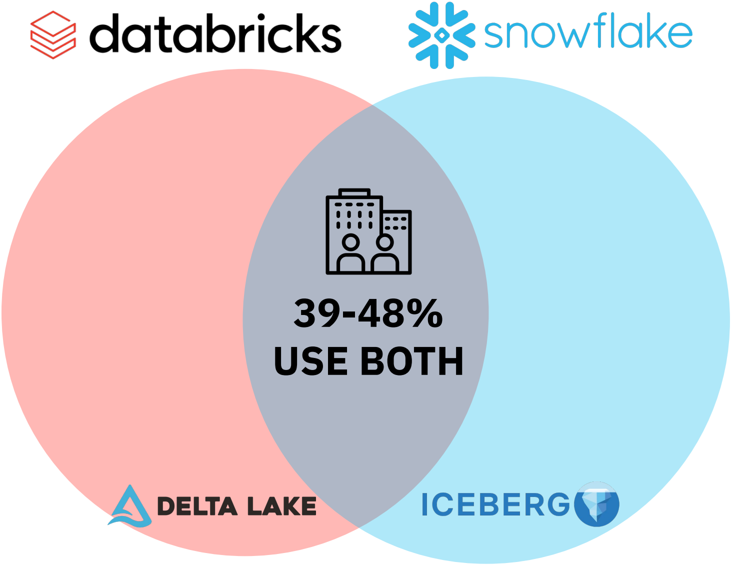
```yaml
sourceFormat: HUDI
targetFormats:
  - DELTA
  - ICEBERG
datasets:
  -
    tableBasePath: s3://path/to/hudi-dataset/people  # replace this with gs://path/to/hudi-dataset/people if yo
    tableName: people
    partitionSpec: city:VALUE
```

```
s3_bucket/my_table/
 |- .hoodie/
 |     |- hoodie.properties
 |     |- metadata/
 |- file_1.parquet
 |- file_2.parquet
 |- file_N.parquet
```

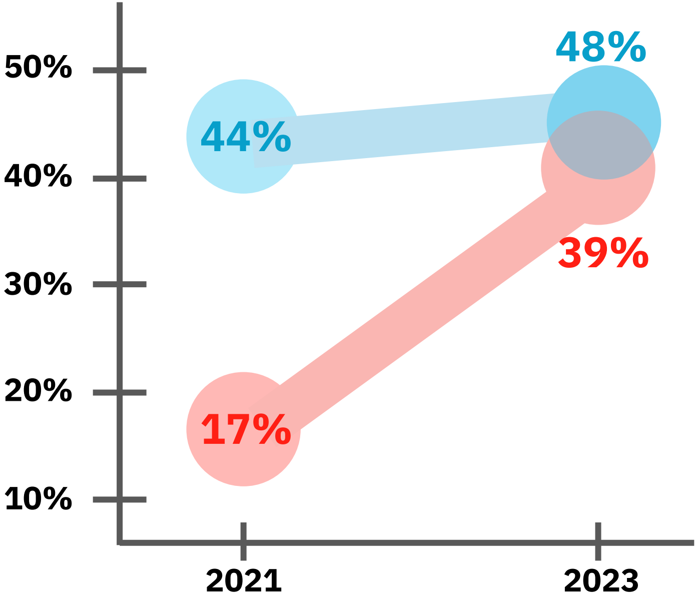
```
s3_bucket/my_table/
 |- .hoodie/
 |     |- hoodie.properties
 |     |- metadata/
 |- _delta_log/
 |     |- 000000.json
 |- metadata/
 |     |- v1.metadata.json
 |     |- snap-9fa1-2-16c3.avro
 |     |- 0d9a-98fa-77.avro
 |- file_1.parquet
 |- file_2.parquet
 |- file_N.parquet
```

```python
# any of these work on the same table
spark.read.format("hudi")
spark.read.format("delta")
spark.read.format("iceberg")
```

A tale of two...



Overlap Growth 2021-2023



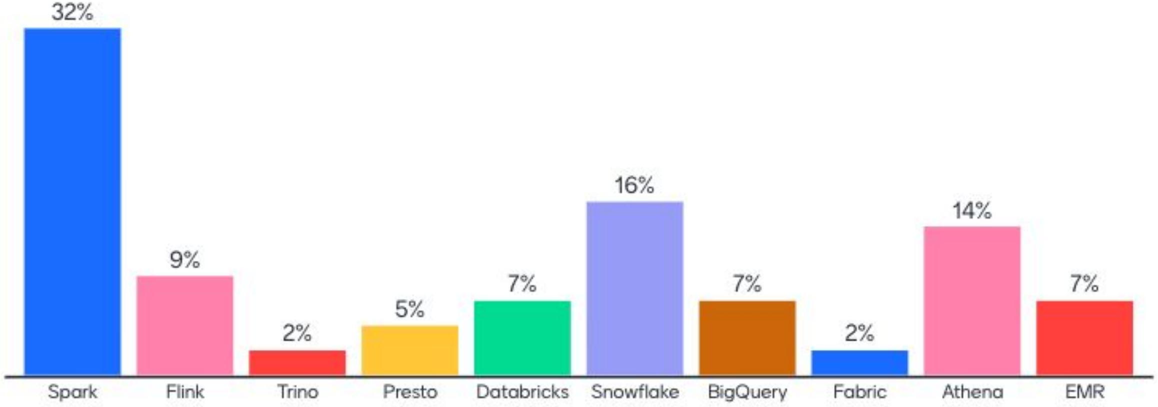
siliconANGLE

https://siliconangle.com/2023/06/30/connecting-dots-snowflakes-data-cloud-ambitions/

Join at menti.com | use code 5780 3382

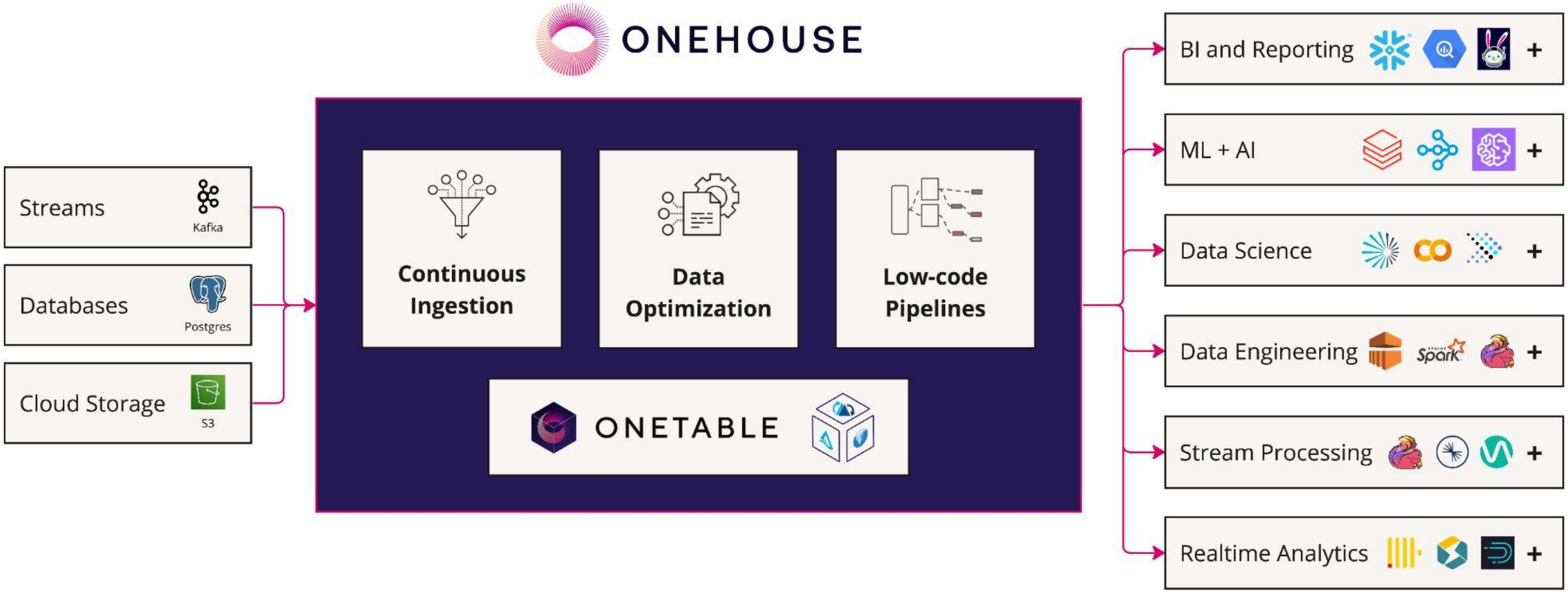
Mentimeter

Which Query Engines Do You Use?



Apache XTable™ - Why Build It?

Whitepaper



Our Goal = Universal Data Lakehouse

Demo Time!

Apache XTable™ - Vision for the Future

Goals

- Seamless and efficient interoperability
- Eliminate data silos
- Project sustainability and evolution

Features

- Real-time and transparent replication in any direction
- Accurate and lossless model
- Extensibility and flexibility

Community

- Neutral and inclusive: Vendors, Cloud providers, Users
- Graduate ASF Incubation

Apache XTable™ - Join the community

Initial Committers

- Tim Brown : ***Onehouse***
- Vamshi Gudavarthi : ***Onehouse***
- Ashvin Agrawal : ***Microsoft***
- Jesus Camacho Rodriguez : ***Microsoft***
- Anoop Johnson : ***Google***
- Stamatis Zampetakis : ***Cloudera***
- Hitesh Shah : ***Adobe***
- Jean-Baptiste Onofré : ***Dremio***
- Baljinder Singh : ***Walmart***
- Vinish Reddy: ***Onehouse***
- Vinoth Chandar: ***Onehouse***

CLOUDERA

Microsoft

ONEHOUSE

Google

Adobe

dremio

Walmart

APACHE INCUBATOR

Apache XTable™ - Roadmap

Current Status

- Supported formats: ***Apache Hudi, Apache Iceberg, and Delta Lake***
- Tested with: Apache Spark, Trino, Microsoft Fabric, Databricks, BigQuery, Snowflake, Redshift, and more
- Features: on-demand incremental conversion, copy-on-write, catalog integration, change-history

Roadmap (6-12 months)

- ***Merge-on-Read*** (delete vectors)
- Apache Paimon (incubating)
- ***Performance***, ***efficiency***, and resiliency
- Deployment: as-a-service and in-memory
- Native engine integration

Roadmap (long term)

- Multi-writer (duplex)
- Synchronized commit timestamp
- Feature parity (superset)
- New technology stack
- Support new formats & versions

Apache XTable™ - Let's Build Together

Github: https://github.com/apache/incubator-xtable

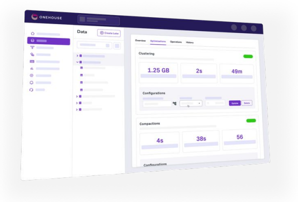
Docs : https://xtable.apache.org/docs/how-to

Twitter : https://twitter.com/apachextable

LinkedIn : https://www.linkedin.com/company/apache-xtable/

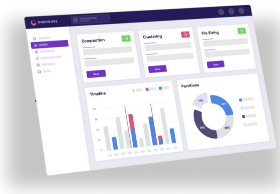
Mailing List : dev-subscribe@xtable.apache.org

ONEHOUSE - 3 Ways to Engage with Onehouse



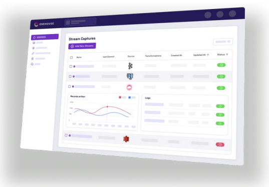
Lakehouse Monitoring - ($0 Free)

- No install, no permissions
- Monitoring and tuning insights for your Hudi, Iceberg, and Delta Lake pipelines



Lakehouse Table Services

- Keep your existing Hudi, Iceberg, Delta pipelines
- Onehouse will automate advanced Table Optimizations for 10x faster analytics



Lakehouse Streaming Ingest/ELT

- 10x Faster/Cheaper vs existing OSS Hudi, Iceberg, Delta pipelines
- Fully managed auto-scaling infrastructure w/ serverless experience in your VPC
- Simple UI + APIs for programmatic and templated CI/CD devops integration

Contact me: kyle@onehouse.ai

Apache Hudi DELTA LAKE ICEBERG

Thank You!