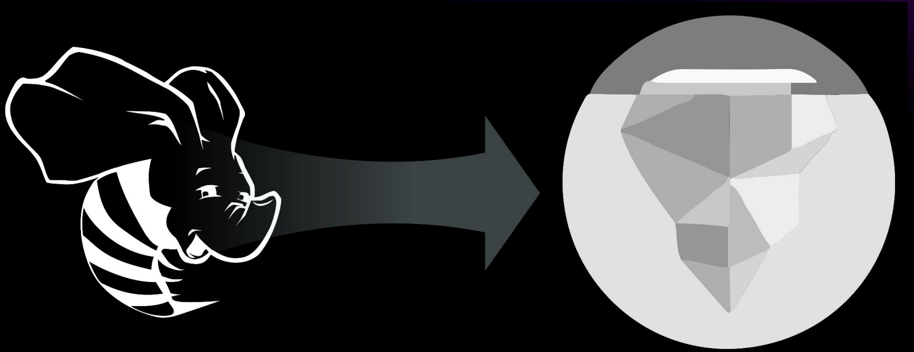
Starburst

Hive to Iceberg

To Migrate, or Not to Migrate

Subtitle Goes Here

Starburst Webinar
May 8, 2024



Connection Before Content

Lester Martin - https://about.me/lestermartin

- Educational Engineer @ Starburst
  - Build the content
  - Teach the class
  - Repeat
- 30 years of technology experience
  - Started my journey on a TRS-80 Model III
  - Played most every role, but consider myself a programmer at my core
  - Half of career in transactional systems and the second half in analytical processing
  - A DECADE of “big data” experience to include
    - Trino/Starburst, Hadoop, Hive, Spark
    - NiFi, Kafka, Storm, Flink
    - HBase, MongoDB

Webinar Agenda

Slides, but DEMOs, too!

- Evolution of a data lakehouse (*the 3 min version*)
- Picking your components
- Building a data lakehouse
- When NOT to migrate from Hive
- Migration strategies
- Additional considerations

Scan for a Trino and Iceberg cheat sheet

Evolution of the data lakehouse

How did we get here?

Starburst

Data Architecture Evolution

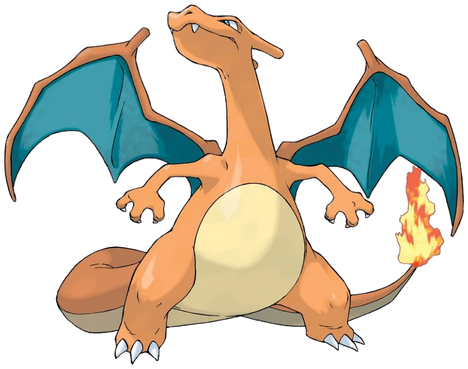
Data Warehouse

Charmander

Data Lake

Charmeleon

Data Lakehouse



Charizard

Starburst

The Data Warehouse

Popularized in the 90’s to provide a 360 degree view

The Good

- Integrates siloed RDBMS’s into one “centralized” location
- Simple & reliable analytical querying
- Data audit, governance and lineage
- Great for small amounts of data

The Bad

- Inability to store unstructured data
- Lack scalability and flexibility
- Tightly coupled storage and compute
- Expensive, proprietary hardware and software (*creating vendor lock-in*)

The Data Lake

Born out of the internet age and big data boom

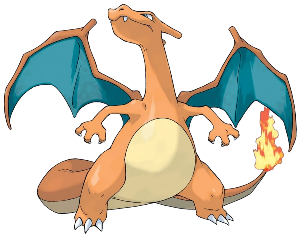
The Good

- In 2006, Apache Hadoop emerges so unstructured data can be processed at a scale previously unimaginable
- Shift toward parallel processing
- Capitalize on low cost object storage
- Allows for greater flexibility (schema on read)

The Bad

- Inability to support transactions, updates, or modifications
- Difficult to get top-tier performance
- Lack of data quality and inconsistent data formats
- Insufficient data lineage and limited data discoverability

The Data Lakehouse



Applying data warehouse principles to the data lake

- Utilize the ***separation of storage and compute*** to apply the reliability, performance, and data quality of the data warehouse to the openness and scalability of the data lake
- ***Increased performance and scalability*** through the use of indexing and caching via your query engine (**Trino**) and modern table formats (ex: **Iceberg**)
- Provide traditional ***data modifications*** (ex: `UPDATE` & `MERGE` commands) with ***ACID transaction*** guarantees over files stored in the data lake
- Tackle ***unstructured, semi-structured, and structured*** analytical data all in a data lakehouse - creating a place for AI/ML & BI use cases alike

Starburst

Picking your components

Trino is the best query engine ever

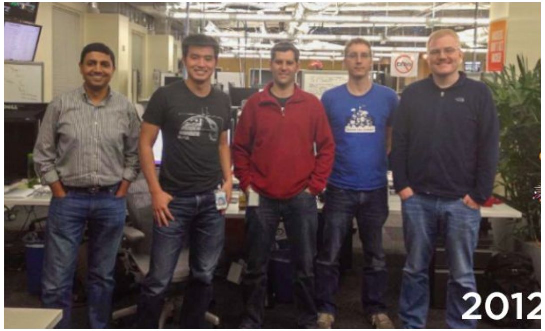
The data accessibility problem

Data practitioners faced the same challenges at Facebook in 2010

- Facebook created Hive to query terabytes of data in Hadoop using SQL
- Data scientists attempted to query massive object stores, but performance was too slow
- Data consumers were limited by the number of queries they could run — often ***fewer than 10*** in one day

Starburst

Enter Trino (Presto)



2012

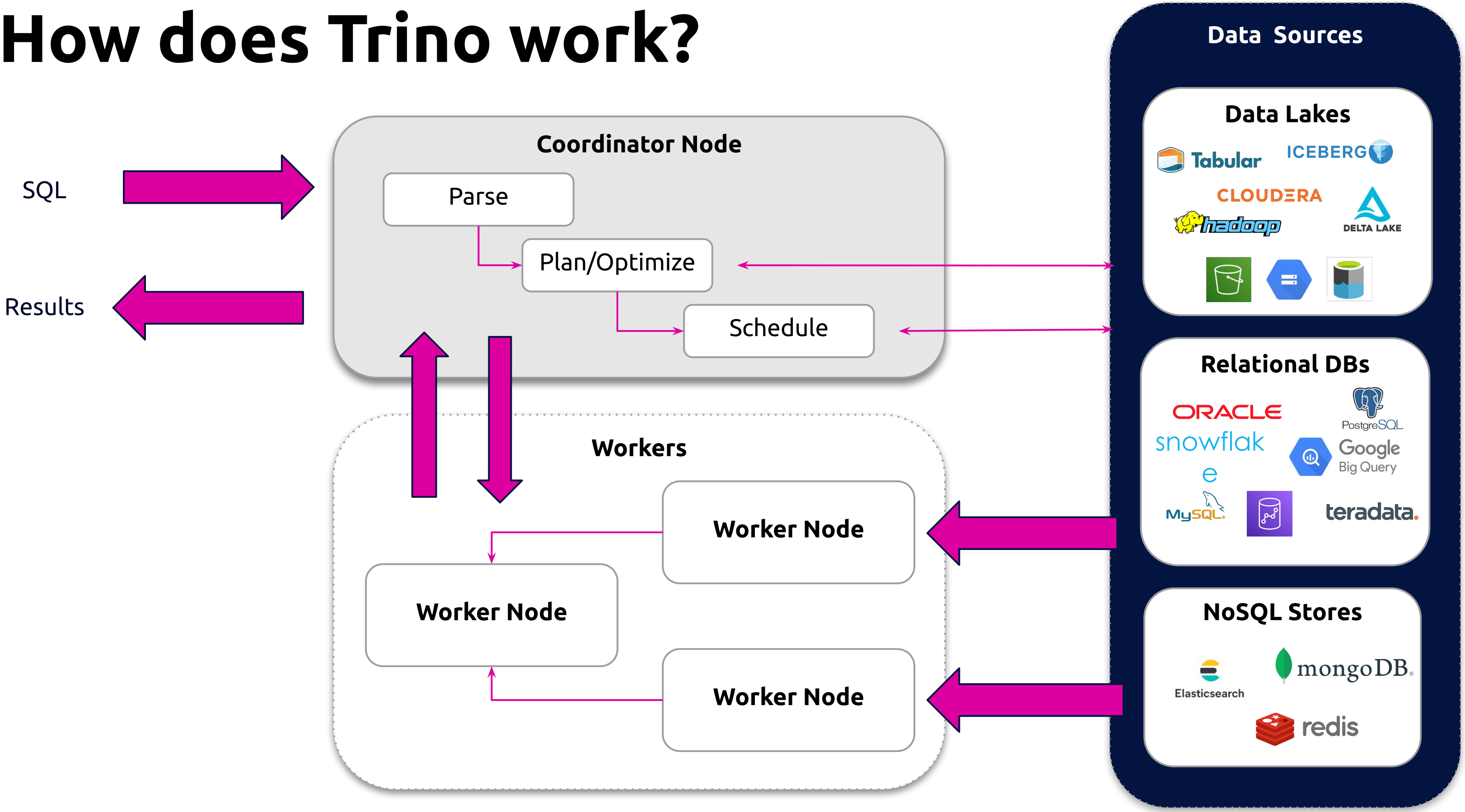
A new open source query engine designed for speed

Trino (*formerly known as Presto*) is a fast distributed SQL query engine designed to query large data sets distributed over one or more heterogeneous data sources.

- Harnesses the power of distributed computing
- Separates compute from storage
- ANSI SQL compliant

https://trino.io

How does Trino work?



SQL

Results

Coordinator Node

Parse

Plan/Optimize

Schedule

Workers

Worker Node

Worker Node

Worker Node

Data Sources

Data Lakes

Relational DBs

NoSQL Stores

https://trino.io

Starburst

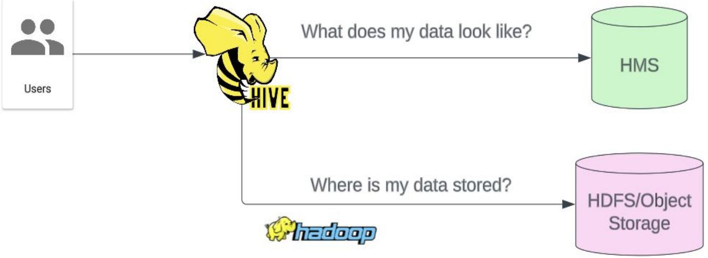
Picking your components

Iceberg is the industry standard table format

The Challenges of the invisible Hive "spec"

Hive has been critical for the evolution of SQL querying in distributed systems

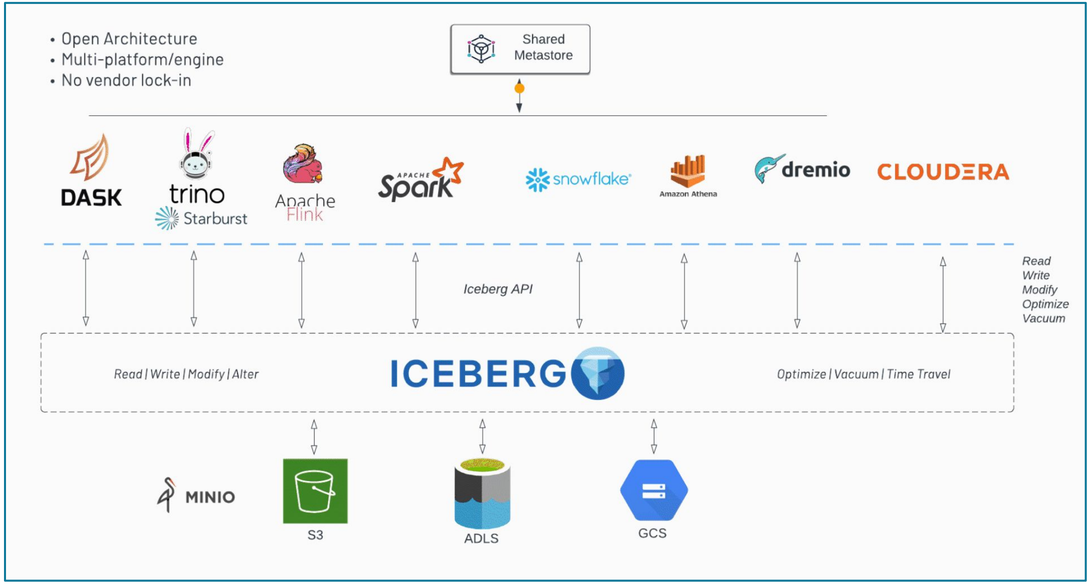
- Rigid partitions – *requires user involvement*
- DIY schema evolution
- Not optimized for object storage – *need to scan all files in a "folder"*
- Bolt-on ACID transactions have always been squirrelly – *inconsistency, correctness issue*
- Performance & scalability concerns with the metastore
- No inherent table content versioning



Apache Iceberg

- Created by Ryan Blue & Daniel Weeks at Netflix in 2017
- Solve the challenges of performance, data modification and schema evolution in the lake + offer benefits of versioning
- Uses open data concepts (orc, parquet, avro) and architecture

Multi-Engine Platform



Iceberg: lake choice + warehouse behavior

SQL behavior

- Schema and partition evolution
- Hidden partitioning

Modern warehouse SQL

- UPDATE / DELETE / MERGE
- Time travel & rollback (via versioning)

Iceberg: lake choice + warehouse behavior

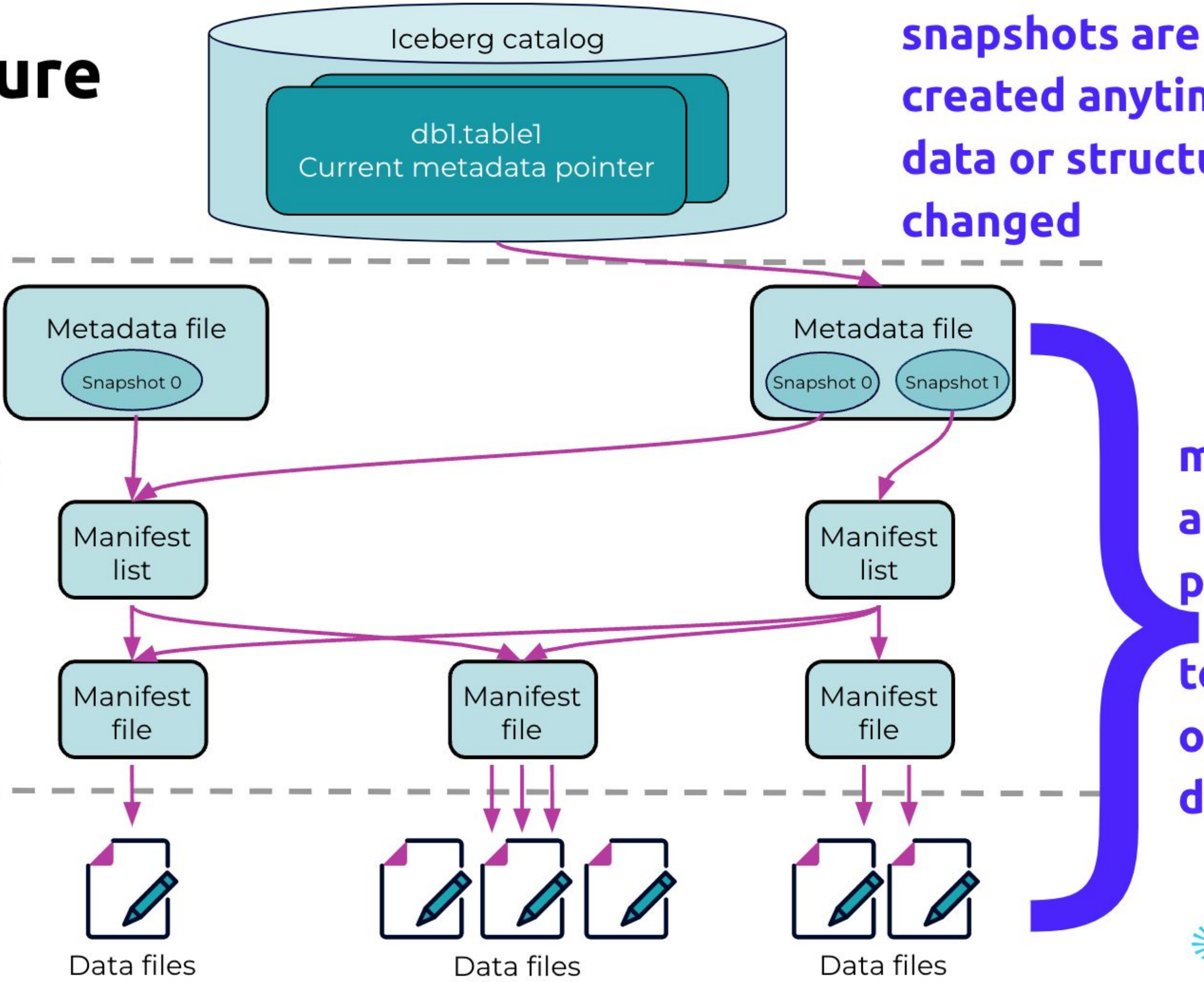
SQL behavior

- Schema and partition evolution
- Hidden partitioning

Modern warehouse SQL

- UPDATE / DELETE / MERGE
- Time travel & rollback (via versioning)

Architecture overview



snapshots are created anytime data or structure is changed

Metadata layer

lives in the `./metadata` dir

saved in the `./data` dir

Data layer

metadata and data persisted together on the data lake

Starburst

Iceberg should be invisible

Avoid unpleasant surprises

- No zombie data
- Performance is not mysterious
- Reduced metastore reliance

Doesn’t steal attention

- Fast metadata operations
- Automate the boring stuff
- Fix problems without migration

Optimistic Concurrency

- Allows multiple writes simultaneously, checks for conflicts before final commit

Universal open standard

Building a data lakehouse

Starburst

Open Data Lakehouse Benefits

Data Warehouse Benefits

- ACID transactions
- Fined grained access control
- Data quality
- High performance and concurrency
- Highly curated data
- *Typically proprietary systems*
- Best for business intelligence use cases

Data Lake Benefits

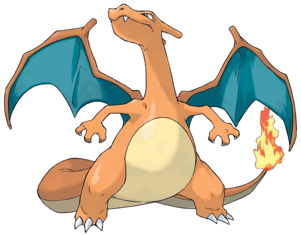
- Petabyte scale
- Cost efficient
- Open formats
- Separation of storage & compute
- Structured and unstructured data
- Best for data science and data engineering use cases

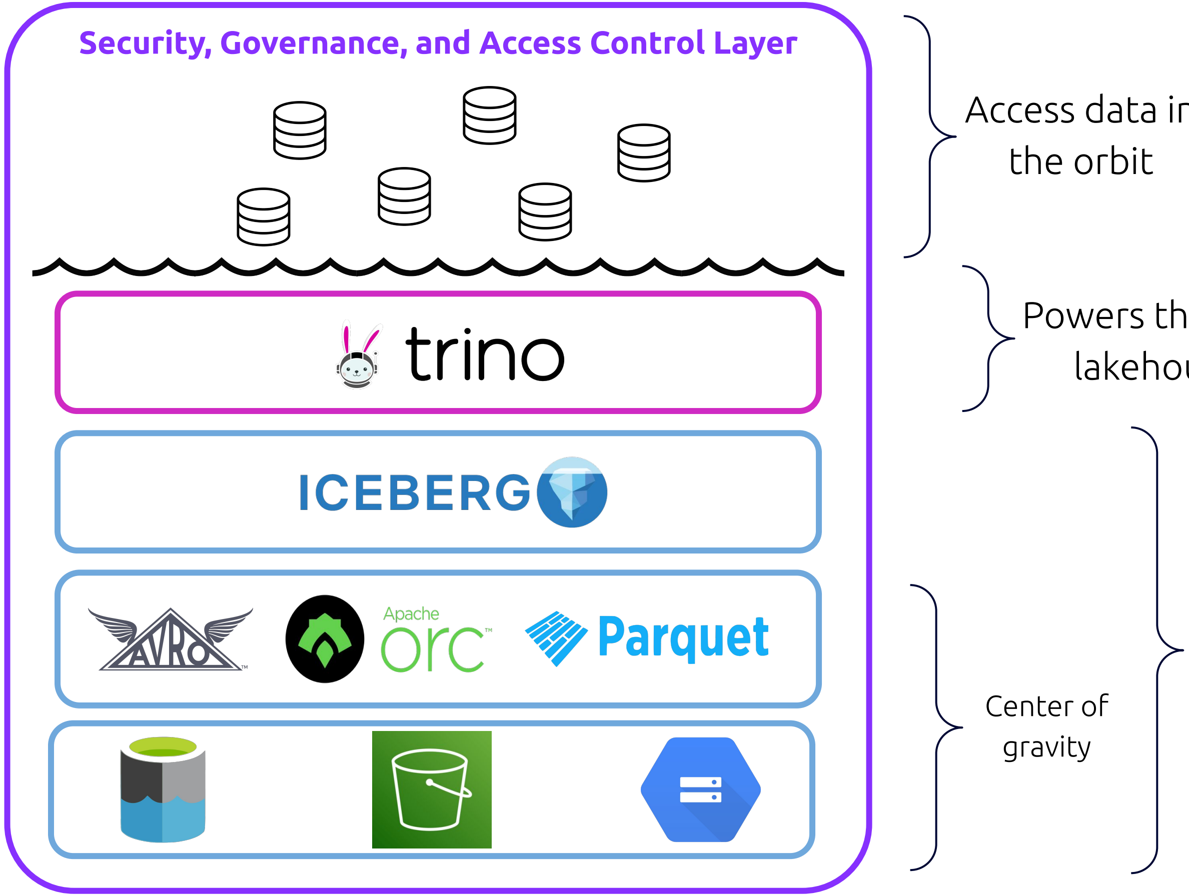
Lakehouse = the doodle of data architecture

Apply data warehouse principles to the data lake of your choice

Starburst

The Open Data Lakehouse





Global federated access to data sources beyond the lake

Compute engine

Table formats

Open file formats

Object storage

Access data in the orbit

Powers the data lakehouse

Center of gravity

Enables data lakehouses

Starburst

When NOT to migrate from Hive

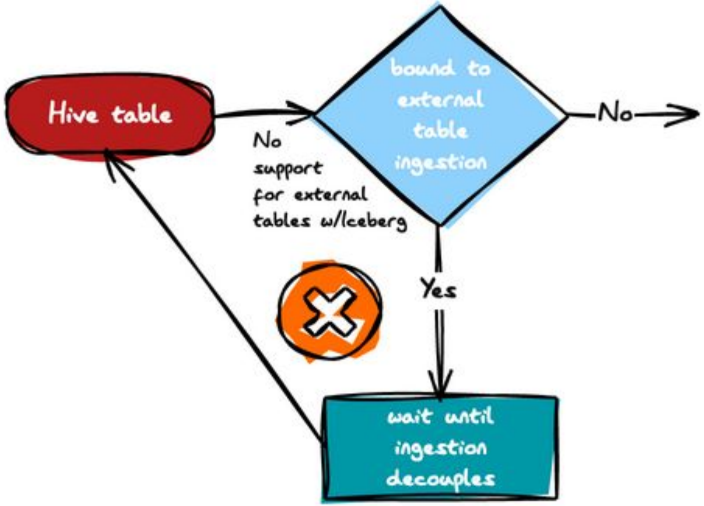
Reasons to NOT migrate

Time and effort will be required - *make sure the juice is worth the squeeze*

Reasons to NOT migrate

Time and effort will be required *- make sure the juice is worth the squeeze*

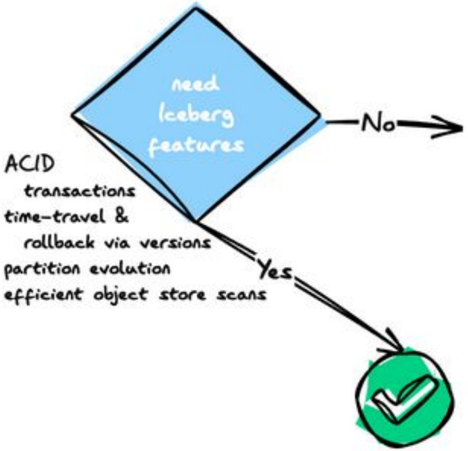
- **Iceberg requires tables to be managed** - no externally ingested files


Hive table
bound to external table ingestion
No
No support for external tables w/Iceberg
Yes
wait until ingestion decouples

Reasons to NOT migrate

Time and effort will be required *- make sure the juice is worth the squeeze*

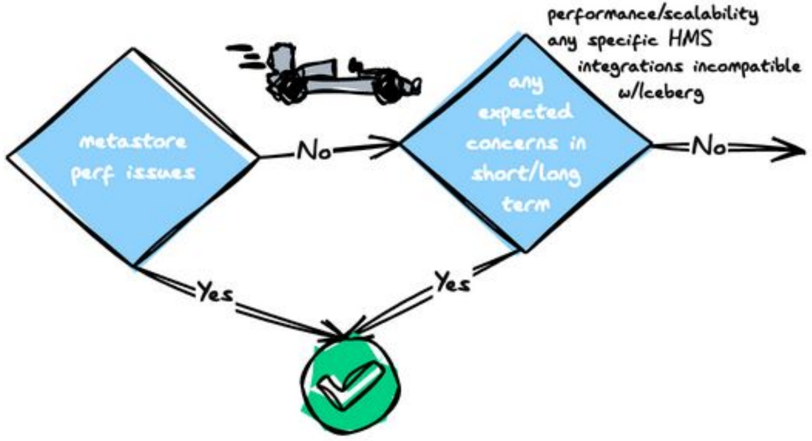
- **Iceberg requires tables to be managed** - no externally ingested files
- **Functional benefits not valuable enough** - ACID transactions, versioning, hidden partitioning, schema/partition evolution



Reasons to NOT migrate

Time and effort will be required *- make sure the juice is worth the squeeze*

- **Iceberg requires tables to be managed** - no externally ingested files
- **Functional benefits not valuable enough** - ACID transactions, versioning, hidden partitioning, schema/partition evolution
- **High degree of confidence in existing performance and scalability** - testing validates no foreseeable concerns in the short to long term



Reasons to NOT migrate

Time and effort will be required *- make sure the juice is worth the squeeze*

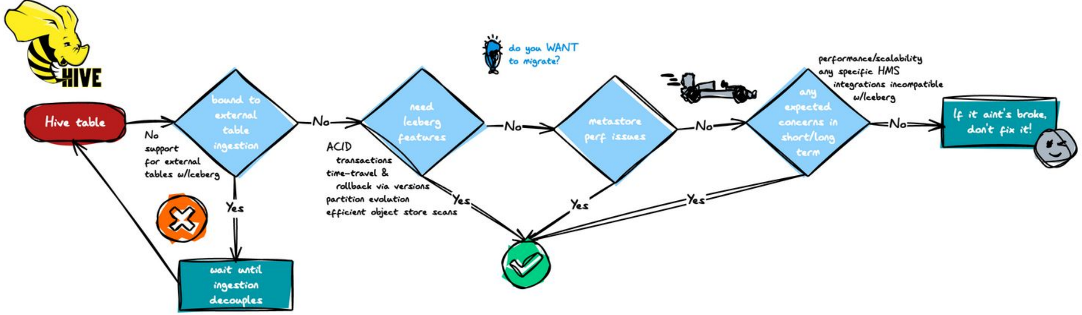
- **Iceberg requires tables to be managed** - no externally ingested files
- **Functional benefits not valuable enough** - ACID transactions, versioning, hidden partitioning, schema/partition evolution
- **High degree of confidence in existing performance and scalability** - testing validates no foreseeable concerns in the short to long term
- **Simply no resources available to migrate** - much less the necessary testing to find any unforeseen issues

Reasons to NOT migrate

do you WANT to migrate?

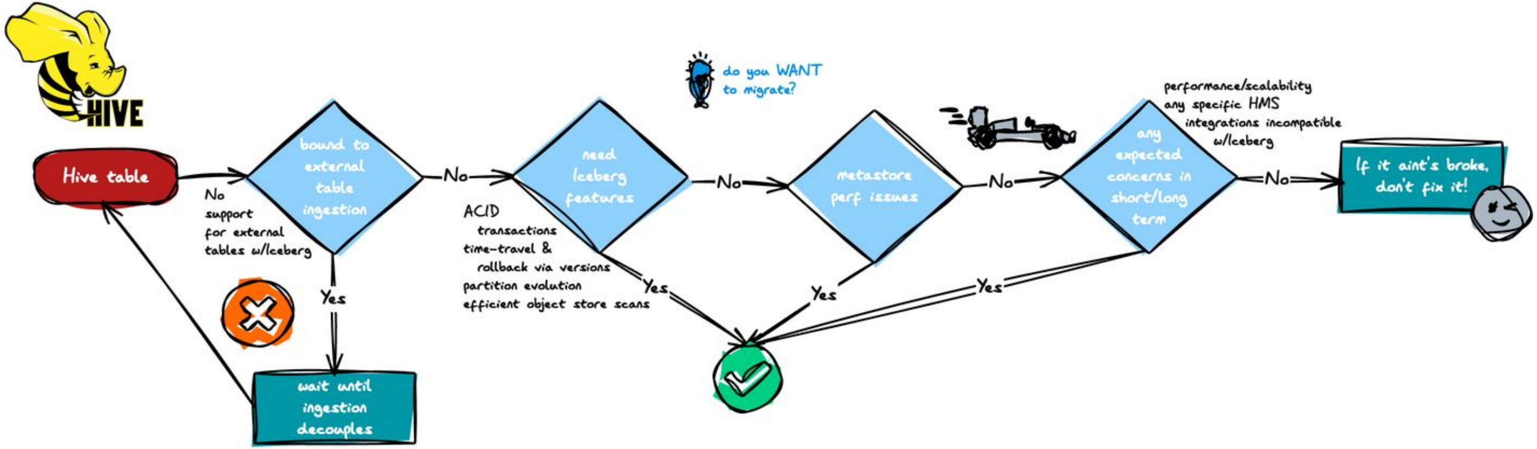
Time and effort will be required *- make sure the juice is worth the squeeze*

- **Iceberg requires tables to be managed** - no externally ingested files
- **Functional benefits not valuable enough** - ACID transactions, versioning, hidden partitioning, schema/partition evolution
- **High degree of confidence in existing performance and scalability** - testing validates no foreseeable concerns in the short to long term
- **Simply no resources available to migrate** - much less the necessary testing to find any unforeseen issues


HIVE
Hive table
No support for external tables w/Iceberg
bound to external table ingestion
Yes
wait until ingestion decouples
No
need Iceberg features
ACID transactions
time-travel & rollback via versions
partition evolution
efficient object store scans
Yes
do you WANT to migrate?
No
metastore perf issues
Yes
No
any expected concerns in short/long term
performance/scalability
any specific HMS integrations incompatible w/Iceberg
Yes
No
If it aint's broke, don't fix it!

Reasons to NOT migrate

Time and effort will be required ***- make sure the juice is worth the squeeze***



Hive table

bound to external table ingestion

No support for external tables w/Iceberg

Yes

wait until ingestion decouples

No

need Iceberg features

ACID transactions time-travel & rollback via versions partition evolution efficient object store scans

Yes

do you WANT to migrate?

No

metastore perf issues

Yes

No

any expected concerns in short/long term

performance/scalability any specific HMS integrations incompatible w/Iceberg

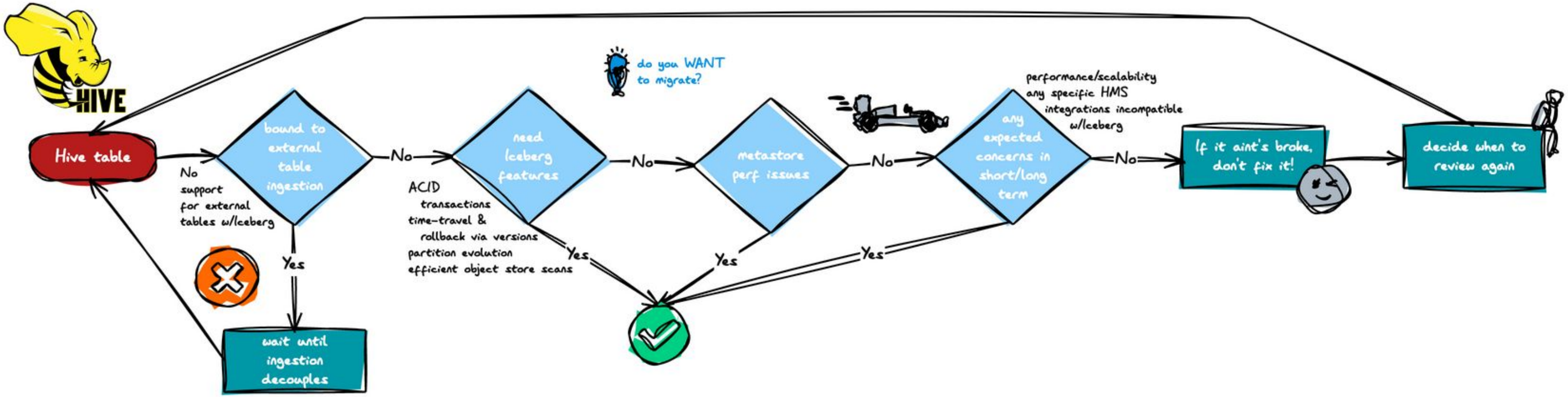
Yes

No

If it aint's broke, don't fix it!

Reasons to NOT migrate (or not yet)

Time and effort will be required - ***make sure the juice is worth the squeeze***



Hive table

No support for external tables w/Iceberg

bound to external table ingestion

Yes

wait until ingestion decouples

No

need Iceberg features

ACID transactions
time-travel & rollback via versions
partition evolution
efficient object store scans

Yes

do you WANT to migrate?

No

metastore perf issues

Yes

No

any expected concerns in short/long term

performance/scalability
any specific HMS integrations incompatible w/Iceberg

Yes

No

If it aint's broke, don't fix it!

decide when to review again

THIS IS HOW WE WOULD GET TO WORK IF WE BELIEVE THE SAYING
"IF IT AIN'T BROKE DON'T FIX IT"
imgflip.com

Migration strategies

In-place vs shadow options

Starburst

Migration strategies

Two approaches - *let's define them*

Shadow migration process

Creates a new Iceberg table modeled after the original Hive table whose values are then inserted into the new table; the original table can then be dropped

The in-place method

Avoids rewriting the data files by modifying the table format type in the catalog and only building additional Iceberg metadata files

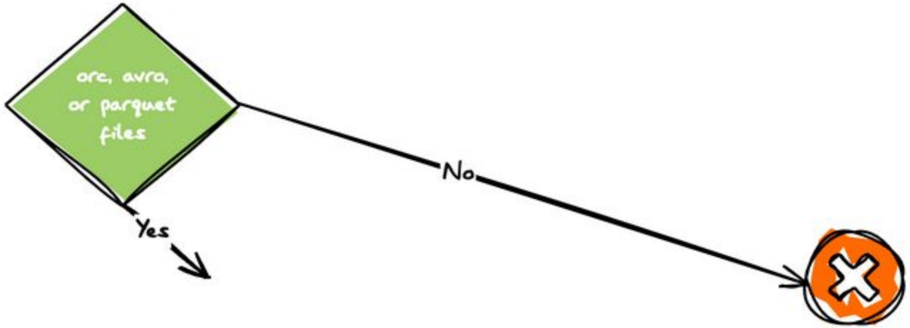
In-place migration requirements

Takes over table's folder & creates Iceberg metadata - *recycles files*

In-place migration requirements

Takes over table's folder & creates Iceberg metadata *- recycles files*

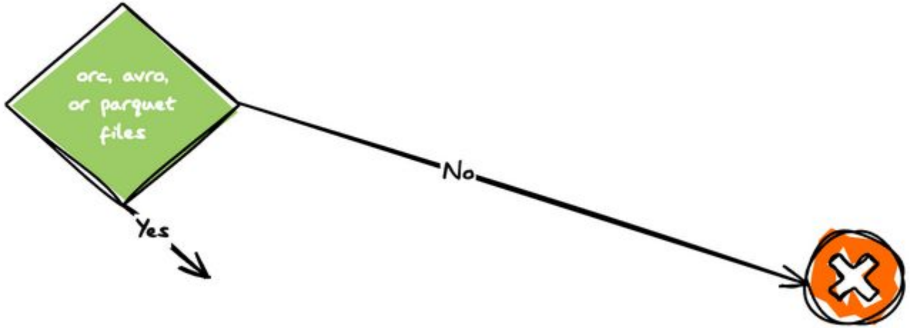
- **Compatible data file formats** - Parquet, ORC & Avro



In-place migration requirements

Takes over table's folder & creates Iceberg metadata ***- recycles files***

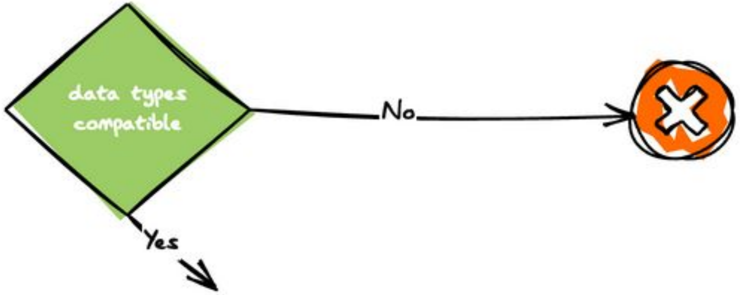
- **Compatible data file formats** - Parquet, ORC & Avro



In-place migration requirements

Takes over table's folder & creates Iceberg metadata *- recycles files*

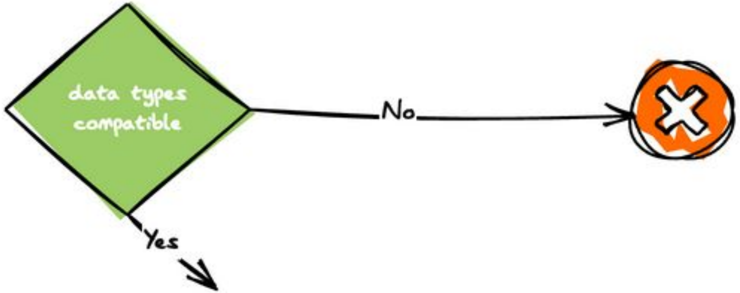
- **Compatible data file formats** - Parquet, ORC & Avro
- **Compatible column data types** - ex: `CHAR` and `TINYINT` not supported and `TIMESTAMP` is not compatible (millisecond vs microsecond precision)



In-place migration requirements

Takes over table's folder & creates Iceberg metadata *- recycles files*

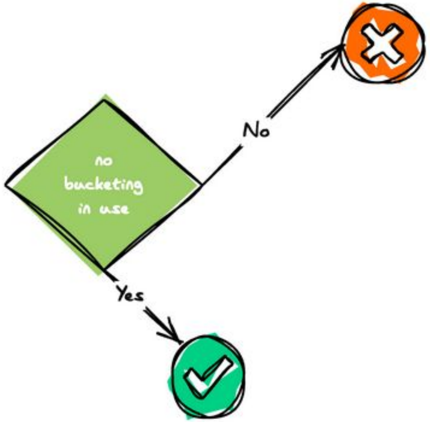
- **Compatible data file formats** - Parquet, ORC & Avro
- **Compatible column data types** - ex: `CHAR` and `TINYINT` not supported and `TIMESTAMP` is not compatible (millisecond vs microsecond precision)



In-place migration requirements

Takes over table's folder & creates Iceberg metadata - *recycles files*

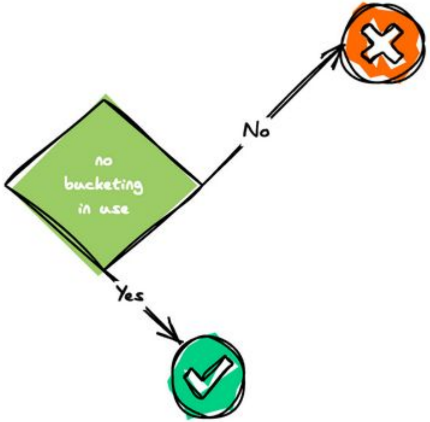
- **Compatible data file formats** - Parquet, ORC & Avro
- **Compatible column data types** - ex: `CHAR` and `TINYINT` not supported and `TIMESTAMP` is not compatible (millisecond vs microsecond precision)
- **`bucketed_by/bucket_count` are ignored** - Iceberg implements bucketing as numbered partitioned folders, not files



In-place migration requirements

Takes over table's folder & creates Iceberg metadata - *recycles files*

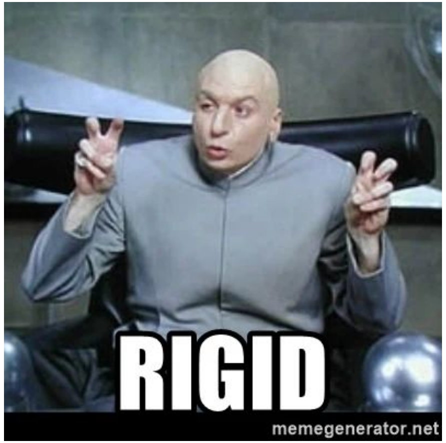
- **Compatible data file formats** - Parquet, ORC & Avro
- **Compatible column data types** - ex: `CHAR` and `TINYINT` not supported and `TIMESTAMP` is not compatible (millisecond vs microsecond precision)
- **`bucketed_by/bucket_count` are ignored** - Iceberg implements bucketing as numbered partitioned folders, not files



In-place migration requirements

Takes over table's folder & creates Iceberg metadata *- recycles files*

- **Compatible data file formats** - Parquet, ORC & Avro
- **Compatible column data types** - ex: `CHAR` and `TINYINT` not supported and `TIMESTAMP` is not compatible (millisecond vs microsecond precision)
- **`bucketed_by/bucket_count` are ignored** - Iceberg implements bucketing as numbered partitioned folders, not files



In-place migration requirements

Takes over table's folder & creates Iceberg metadata *- recycles files*

- **Compatible data file formats** - Parquet, ORC & Avro
- **Compatible column data types** - ex: `CHAR` and `TINYINT` not supported and `TIMESTAMP` is not compatible (millisecond vs microsecond precision)
- **`bucketed_by/bucket_count` are ignored** - Iceberg implements bucketing as numbered partitioned folders, not files

In-place migration requirements

Takes over table's folder & creates Iceberg metadata *- recycles files*

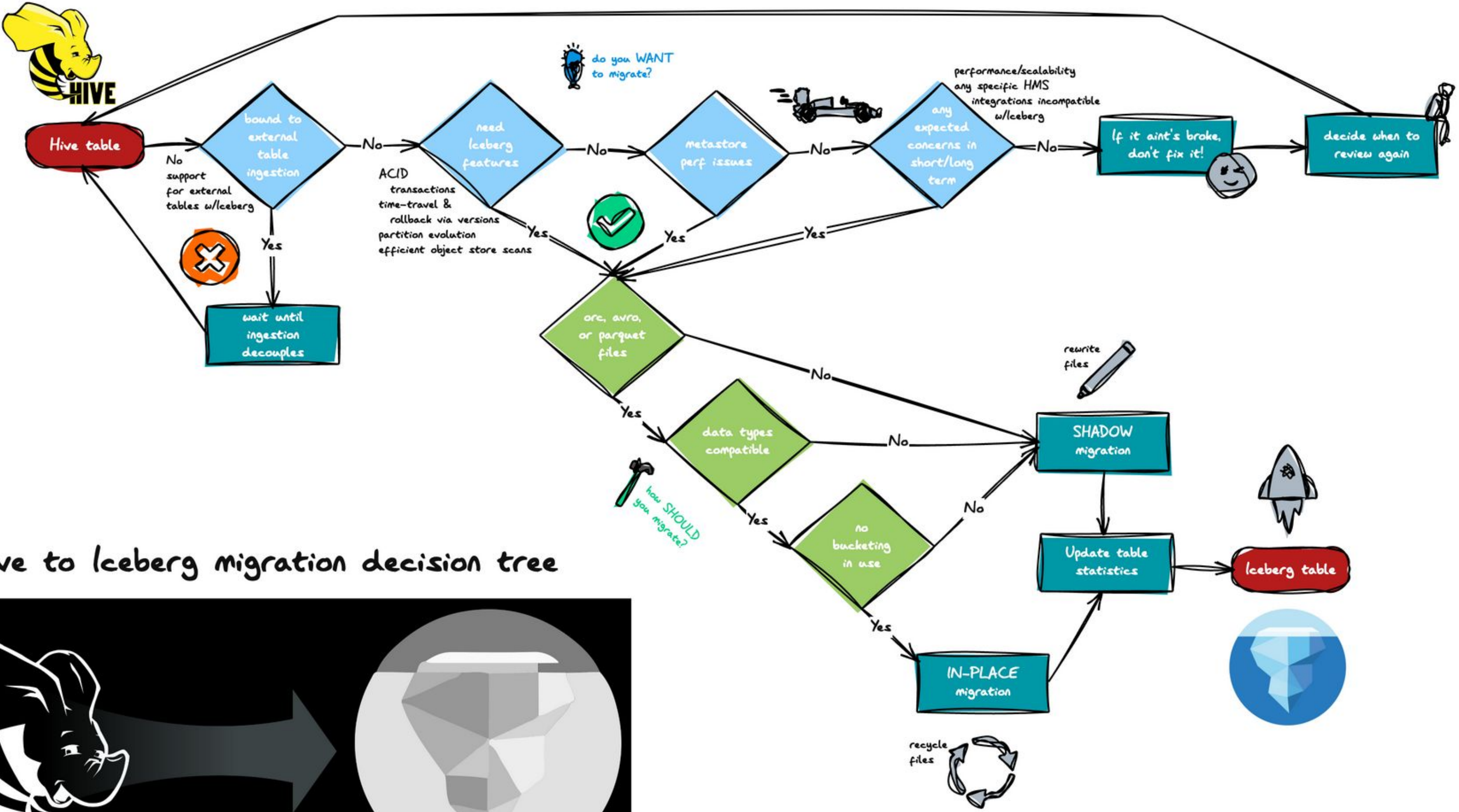
- **Compatible data file formats** - Parquet, ORC & Avro
- **Compatible column data types** - ex: `CHAR` and `TINYINT` not supported and `TIMESTAMP` is not compatible (millisecond vs microsecond precision)
- **`bucketed_by/bucket_count` are ignored** - Iceberg implements bucketing as numbered partitioned folders, not files

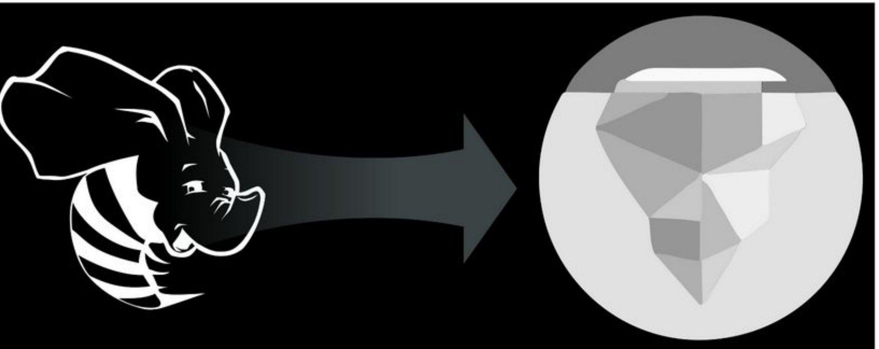
IT WORKS

Put it all together

Starburst

Hive to Iceberg migration decision tree


HIVE
Hive table
bound to external table ingestion
No support for external tables w/Iceberg
Yes
wait until ingestion decouples
No
need Iceberg features
ACID transactions
time-travel & rollback via versions
partition evolution
efficient object store scans
do you WANT to migrate?
No
metastore perf issues
No
any expected concerns in short/long term
performance/scalability
any specific HMS integrations incompatible w/Iceberg
No
If it aint's broke, don't fix it!
decide when to review again
Yes
Yes
Yes
orc, avro, or parquet files
No
Yes
data types compatible
No
Yes
no bucketing in use
No
Yes
how SHOULD you migrate?
rewrite files
SHADOW migration
Update table statistics
Iceberg table
IN-PLACE migration
recycle files



Additional considerations

Migration considerations

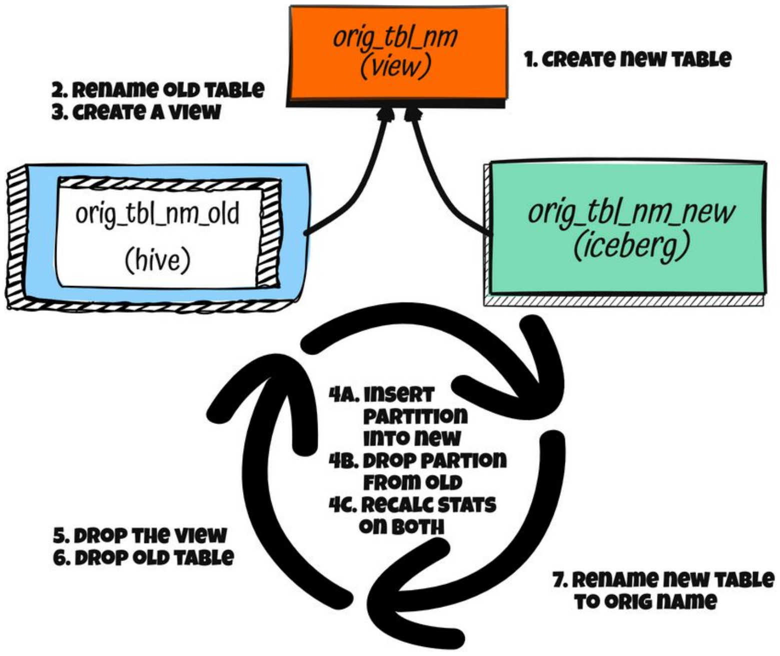
Any migration is an major event and should be planned & tested accordingly

- Test, test, and more test - *don’t forget about a backout strategy*
- Automate maintenance activities
- Consider staging rewrites for very large, heavily-partitioned, tables

Migration considerations

Any migration is an major event and should be planned & tested accordingly

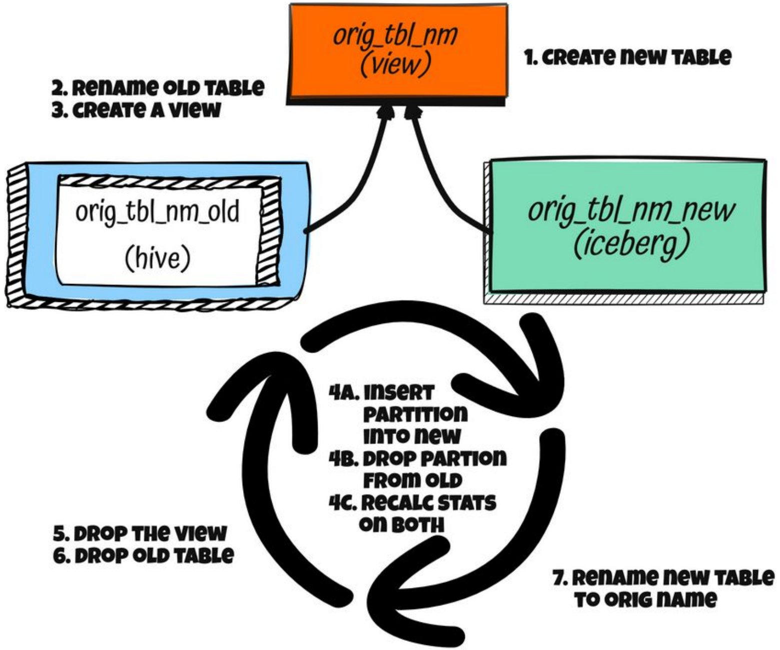
- Test, test, and more test - *don't forget about a backout strategy*
- Automate maintenance activities
- Consider staging rewrites for very large, heavily-partitioned, tables



Migration considerations

Any migration is an major event and should be planned & tested accordingly

- Test, test, and more test - *don't forget about a backout strategy*
- Automate maintenance activities
- Consider staging rewrites for very large, heavily-partitioned, tables



If time permits

Next steps

What are my next steps?

Decide if you are ready to begin & then engage Starburst for help

- Evaluate your existing data lake tables
- Consider tactical focus on largest tables vs comprehensive migration of all
- Visit https://www.starburst.io/solutions/data-migrations/hive-iceberg/ for more information
- Get free guidance on your Hive to Iceberg migration by providing contact info at https://www.starburst.io/info/hive-to-iceberg-migration-guidance/

Demo artifacts at https://github.com/lestermartin/events/tree/main/2024-05-08_Hive2Iceberg

Starburst

Thank you!

Starburst