GNARLY Data_Waves

PRESENTED BY dremio

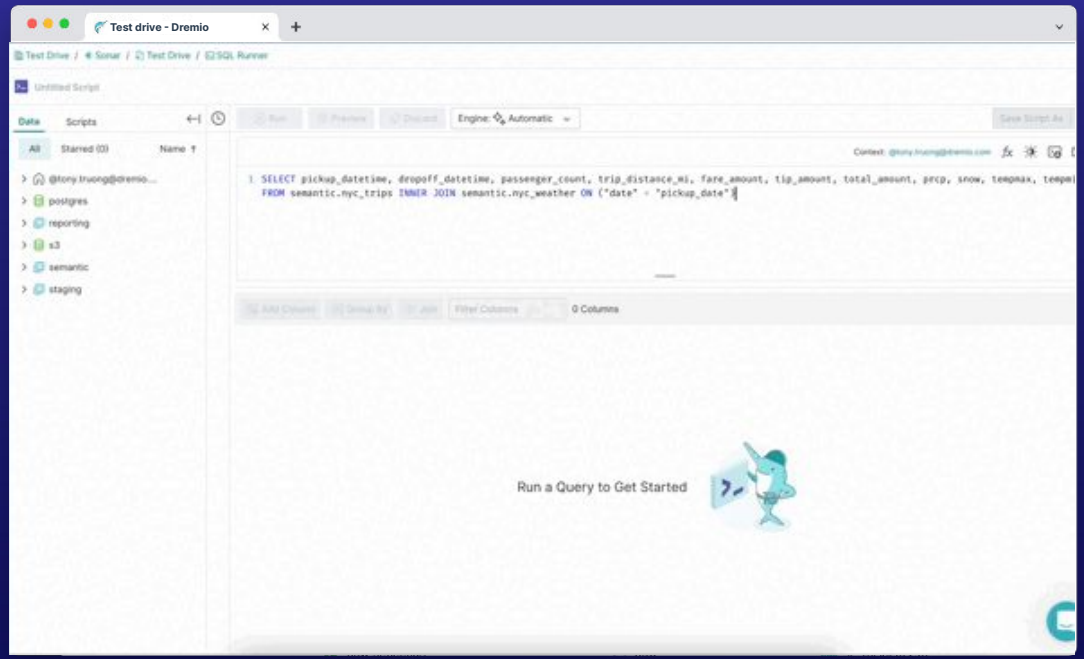
EPISODE 12

How to Migrate Hive to the Data Lakehouse with Dremio and Apache Iceberg

Experience the data lakehouse with Dremio Test Drive

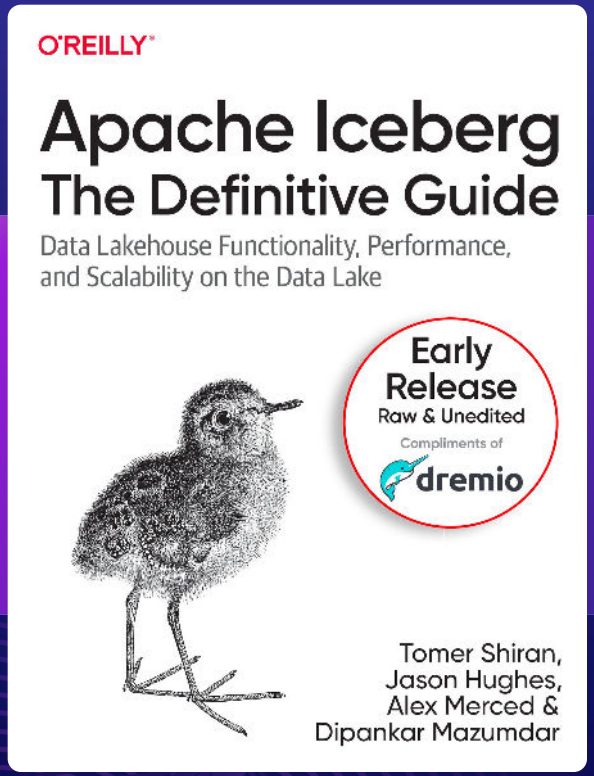
- Sub-second query on 1 billion rows of data joining Amazon S3 with a Postgres database
- Connect to Tableau or Power BI and build a dashboard with this dataset
- Everything hosted by Dremio - 100% free for you

Start Test Drive



GNARLY Data_Waves

Apache Iceberg: The Definitive Guide



Upcoming shows

Register now

EPISODE 13

GDW: Making the Move: Five Factors to Consider When Migrating from Hadoop to the Data Lakehouse

April 18, 2023 at 8AM PST | 11AM EST | 4PM GMT

EPISODE 14

GDW: Enabling data mesh with Dremio Arctic and Data as Code

April 25, 2023 at 8AM PST | 11AM EST | 4PM GMT

EPISODE 15

TBA

May 2nd, 2023 at 8AM PST | 11AM EST | 4PM GMT

EPISODE 15

TBA

May 2nd, 2023 at 8AM PST | 11AM EST | 4PM GMT

How to Migrate Hive to the Data Lakehouse with Dremio and Apache Iceberg

Alex Merced

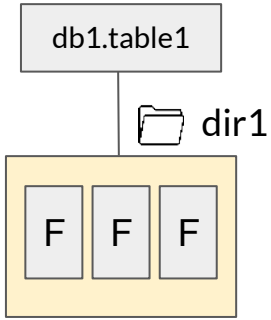
Developer Advocate, Dremio

GNARLY Data_Waves

What is a table format?

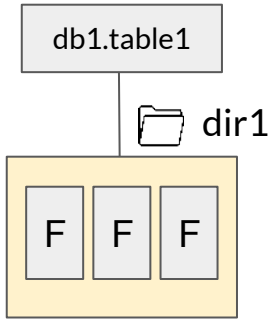
- A way to organize a dataset's files to present them as a single "table"
- A way to answer the question "what data is in this table?"

Old way (Hive)



A table's contents is all files in that table's directories

Hive Table Format



- A table's contents is all files in that table's directories
- The old de-facto standard

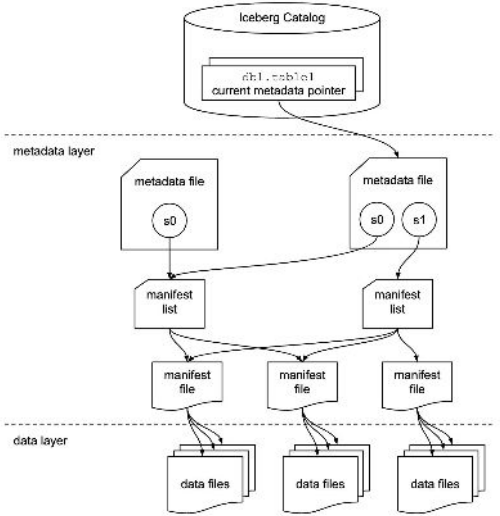
Pros

- Works with basically every engine since it's been the de-facto standard for so long
- More efficient access patterns than full-table scans for every query
- File format agnostic
- Atomically update a whole partition
- Single, central answer to "what data is in this table" for the whole ecosystem

Cons

- Smaller updates are very inefficient
- No way to change data in multiple partitions safely
- In practice, multiple jobs modifying the same dataset don't do so safely
- All of the directory listings needed for large tables take a *long* time
- Users have to know the physical layout of the table
- Hive table statistics are often stale

What is Apache Iceberg?



Apache Iceberg's approach is to define the table through three layers of metadata. These categories are:

- metadata files that define the table
- manifest lists that define a snapshot of the table, with a list of manifests that make up the snapshot and metadata about their data
- manifests is a list of data files along with metadata on those data files for file pruning

Apache Iceberg

GNARLY Data Waves

Why Migration to Apache Iceberg

Features:

- Partition evolution: Ability to update tables partitioning scheme without rewriting table.
- Compatibility Mode: Be able to distribute data in one partition across many prefixes to avoid object storage throttling.
- Hidden Partitioning: Makes partitioning easy to use for consumers so they don't need intimate knowledge of each tables engineering for performant queries.

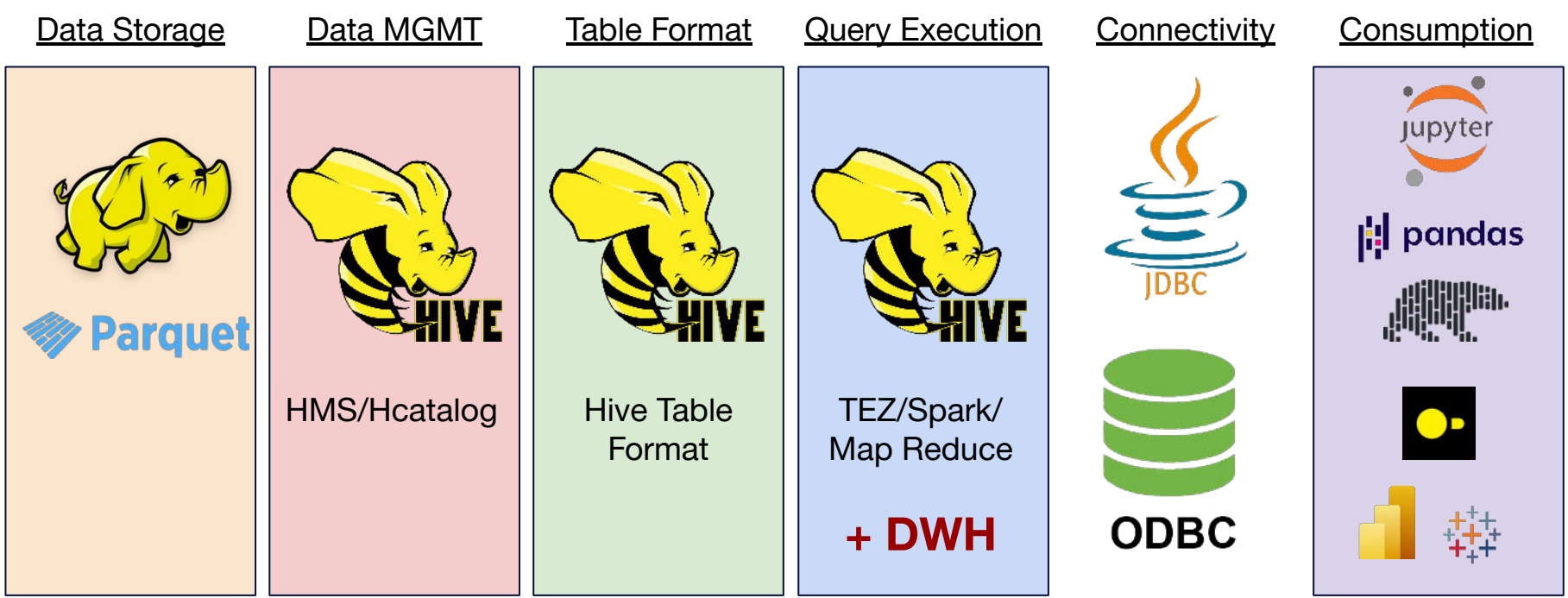
Tooling:

- Read/Write Support: Very extensive read/write support across query engines.
- Data as Code: Make use of git like isolation for mistake recovery and governance with Project Nessie based catalogs like Dremio Arctic.

Openness:

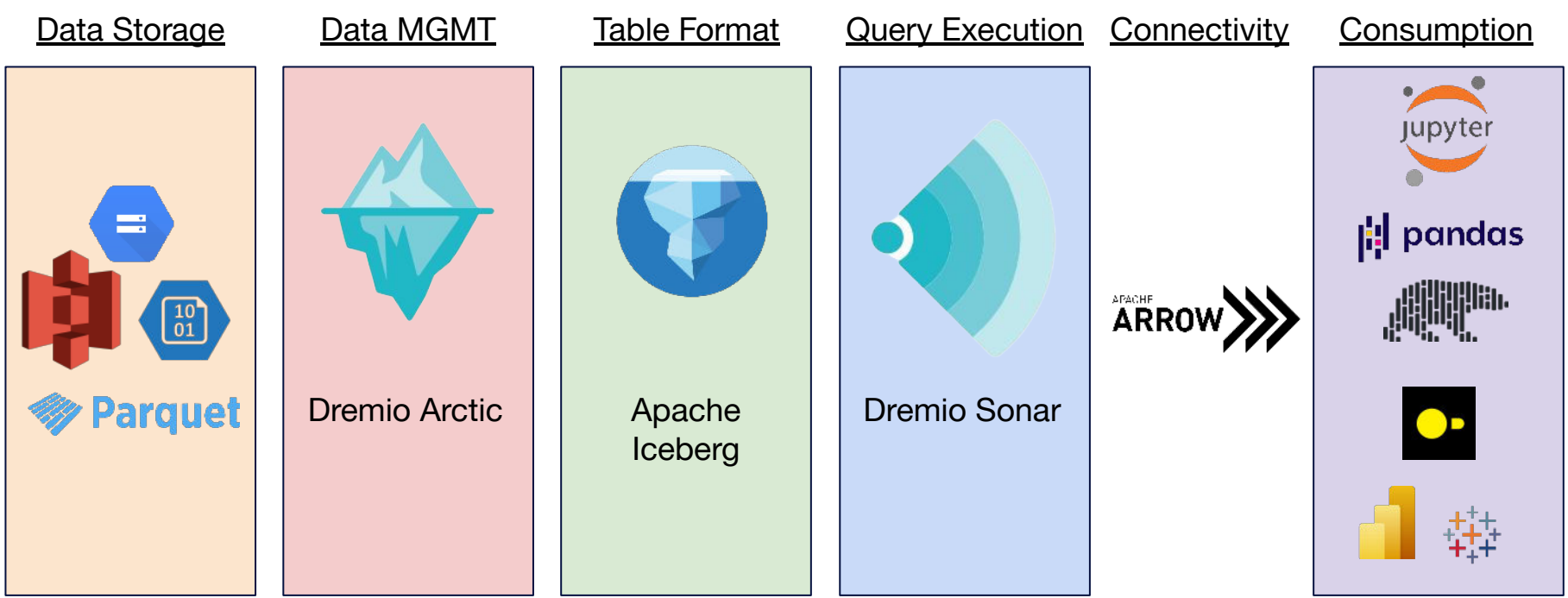
- Transparent project management (single public repo, public meetings, mailing list)
- Wide variety of prominent contributors (Netflix, Apple, Dremio and more)
- Active slack channel for development and usage support

Where the Journey Begins



Slower, harder, and pricier

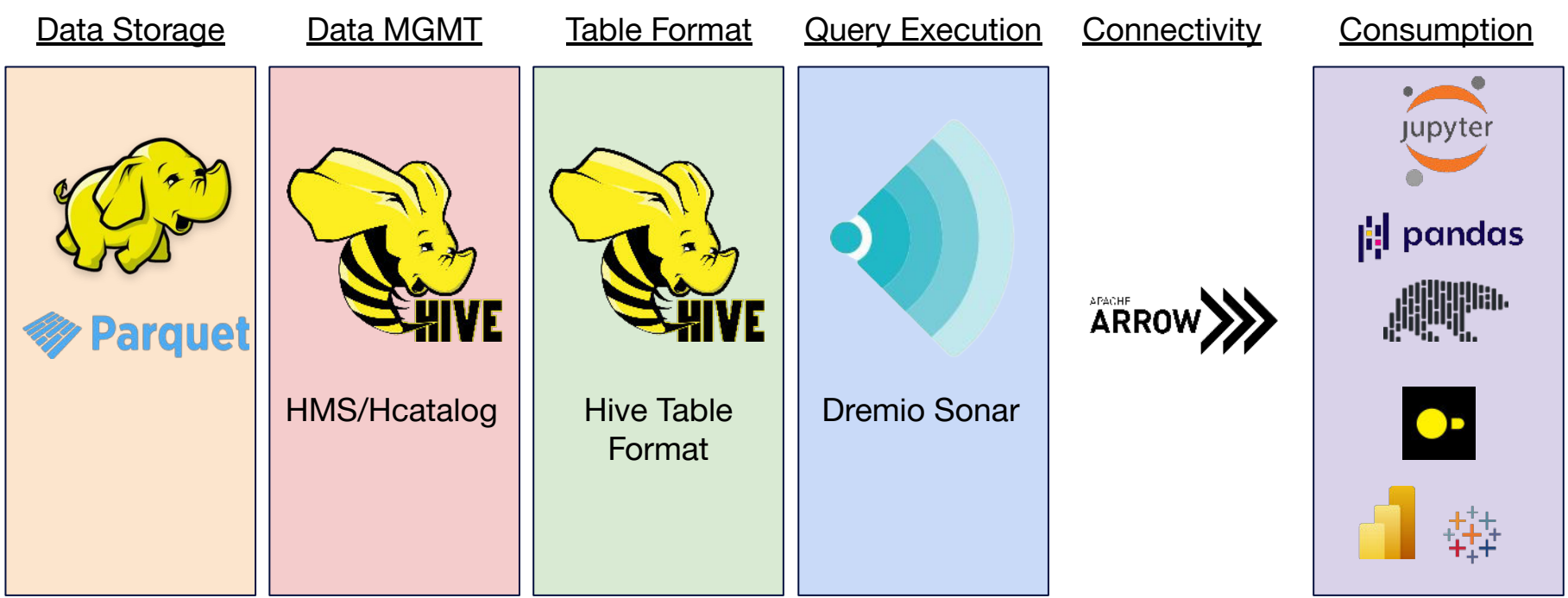
Where the Journey Ends



Faster, Easier and Affordable

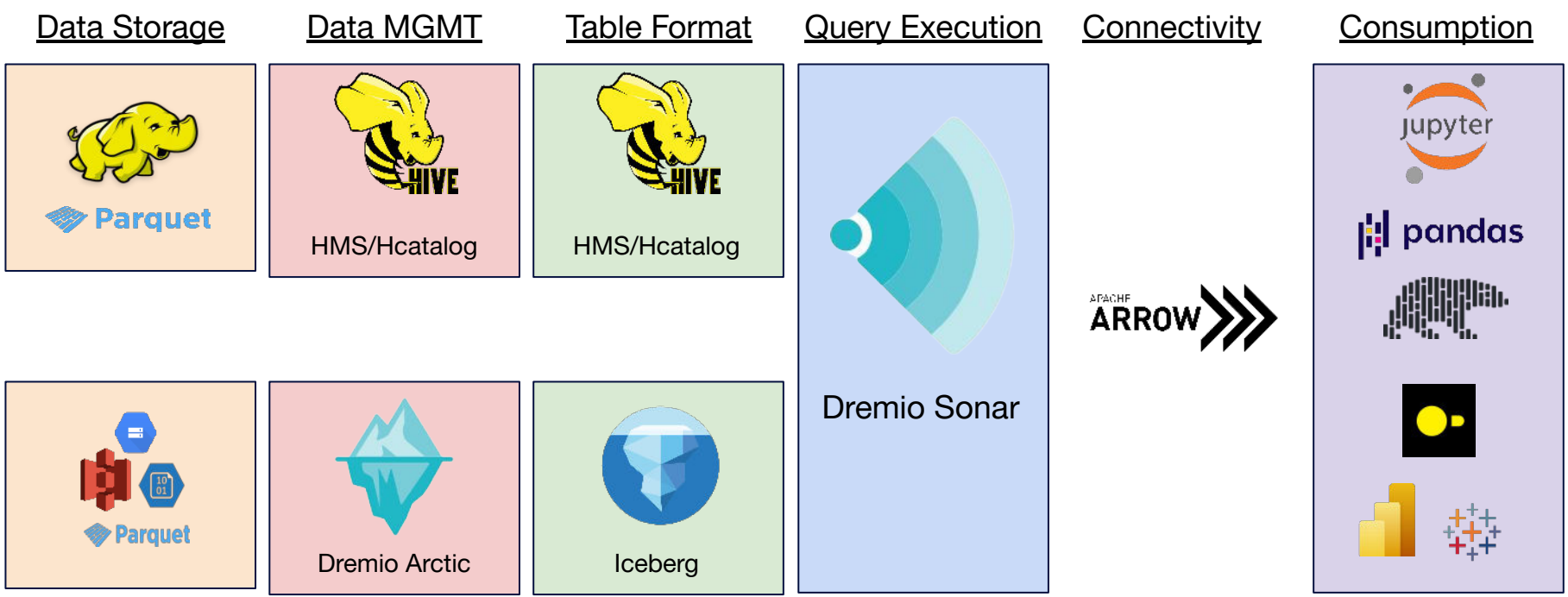
GNARLY Data_Waves

The Road There Part 1

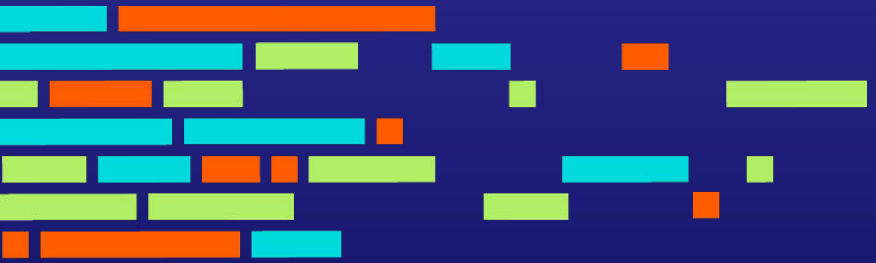


First We Establish The User Workflow

The Road There Part 2



That Interface Becomes the access point to data in the new world and the old world



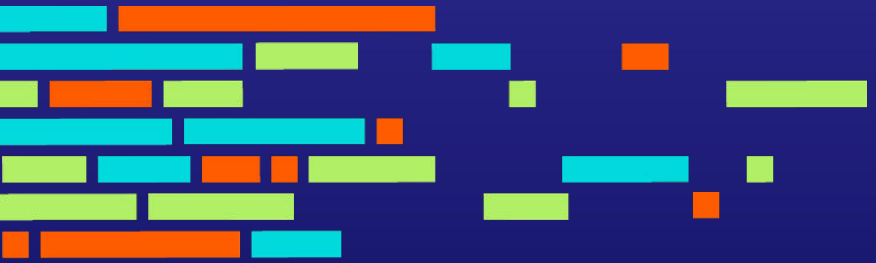
Choosing your Iceberg Catalog

The First step to migrate

Choosing Your Iceberg Catalog

Type of Catalog	Pros	Cons
Project Nessie	• Git Like Functionality • Cloud Managed Service (Arctic)	• Support from engines beyond Spark, Dremio, Flink and Presto • If not using arctic must deploy and maintain Nessie server
Hive Metastore	• Can use existing Hive Metastore	• You have to deploy and maintain a hive metastore
AWS Glue	• Interop with AWS Services	• Support outside of AWS ecosystem and tools that support it

- HDFS, JDBC and REST catalogs also available
- Can use the registerTable catalog method to register a table from one catalog to another



Two Approaches to Migration

Shadow and In-Place Migrations

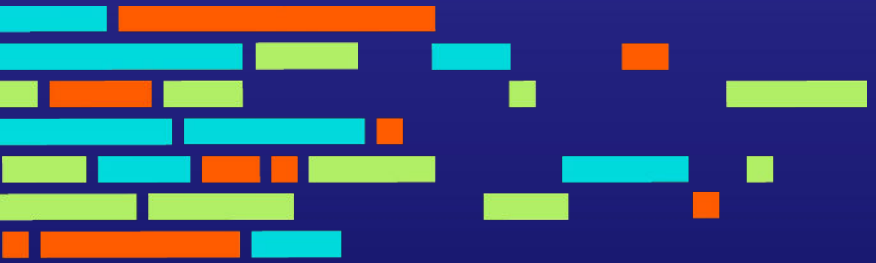
Two Approaches to Iceberg Migration

In-Place Migration

Use the parquet files of a table in a different format and use them for a new or existing Iceberg table.
No need to rewrite all files but schema and partitioning stays the same.

Shadow Migration

Rewrite all the data into a new Iceberg table using a CTAS statement.
Can apply optimizations during rewrite suchs as partitioning, sorting and schema updates but must rewrite all data.



How to do an In-Place Migration

Migrate without Rewriting Your Data

Existing Parquet files that are part of a Hive Table

Apache Iceberg has a Call procedure called **migrate** which can be used to create a new Iceberg table from an the existing Hive table. This uses the existing data files in the table (must be parquet). This REPLACES the Hive table with an Iceberg table, so the Hive table will no longer exist after this operation:

```
CALL catalog_name.system.migrate('db.sample')
```

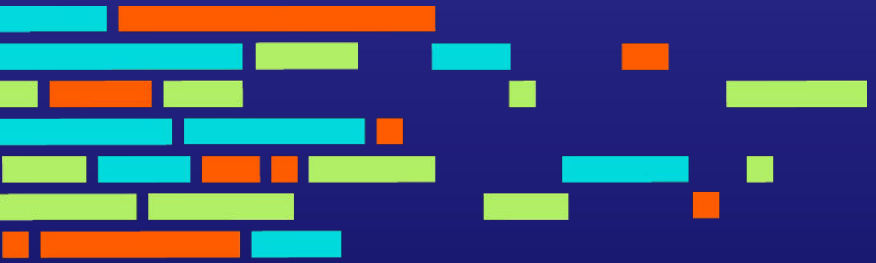
To test the results of migrate before running migrate you can use the **snapshot** call procedure to create temporary iceberg table based on a Hive table without replacing the Hive table. (temporary tables have limitations).

```
CALL catalog_name.system.snapshot('db.sample', 'db.snap')
```

Parquet files not part of a hive table

If you have paquet files representing a dataset that are not yet part of an Iceberg table, you can use the **add_files** procedure to add those datafiles to that of an existing table with a matching schema.

```
CALL spark_catalog.system.add_files(
 table => 'db.tbl',
 source_table => '`parquet`.`path/to/table`'
)
```



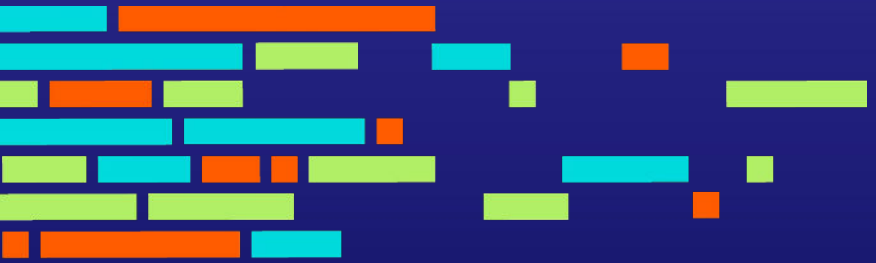
How to Do a Shadow Migration

What reasons would you want to migrate?

Migration by Restating the Data

You can use **CREATE TABLE...AS (CTAS)** statements to create new Iceberg tables from pre-existing tables. This will create new data files but allows you to modify partitioning, schema and do validations at the time of migration if choose to. Since the data is being rewritten/restated, this will be more time consuming that **migrate** or **add_files**. (This can be done with Spark or Dremio)

```
CREATE TABLE prod.db.sample
USING iceberg
PARTITIONED BY (bucket(16, id), days(ts), truncate(last_name, 2))
AS SELECT ...
```



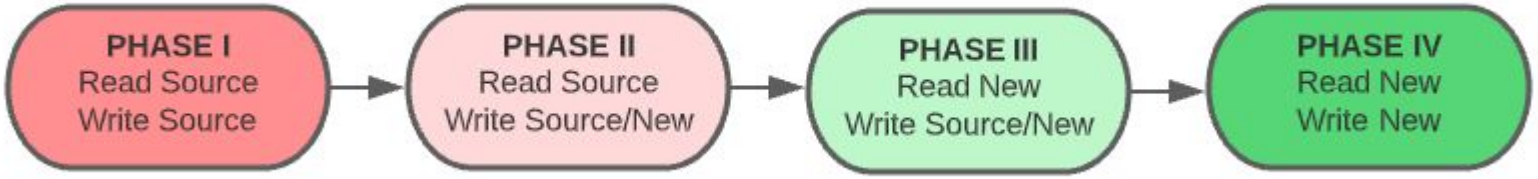
Best Practices

What reasons would you want to migrate?

When to use which approach?

Approach	Are Data Files Rewritten	Can I update Schema and Partition while migrating?	When to use?
migrate	No	No	Moving existing hive table to a NEW iceberg table
add_files	No	No	Moving files from existing table in parquet to an existing Iceberg table
CTAS	Yes	Yes	Moving existing table anywhere to NEW iceberg table with schema or partitioning changes
registerTable (CLI tool coming soon)	No	No	Moving Iceberg table from one catalog to another.

Best Practices

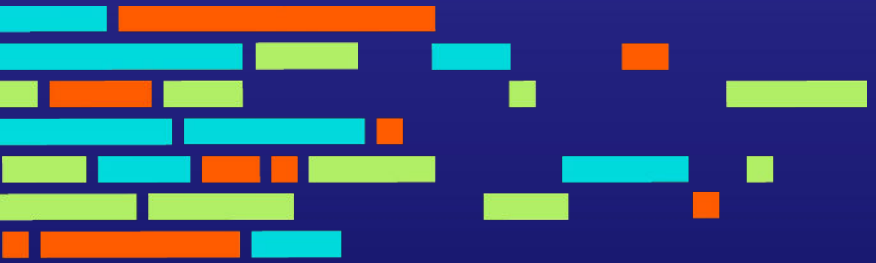


1. As the process begins, the new Iceberg table is not yet created or in sync with the source. User-facing read and write operations remain operating on the source table.

2. The table is created but not fully in sync. Read operations are applied on the source table and write operations are applied to the source and new table.

3. Once the new table is in sync, you can switch to read operations on the new table. Until you are certain the migration is successful, continue to apply write operations to the source and new table.

4. When everything is tested, synced, and working properly, you can apply all read and write operations to the new Iceberg table and retire the source table



Thank You

Follow me @amdatalakehouse