



# Trino on Ice: Using Iceberg to replace the Hive table format

Brian Olsen

Developer Advocate @  Starburst

Trino Contributor



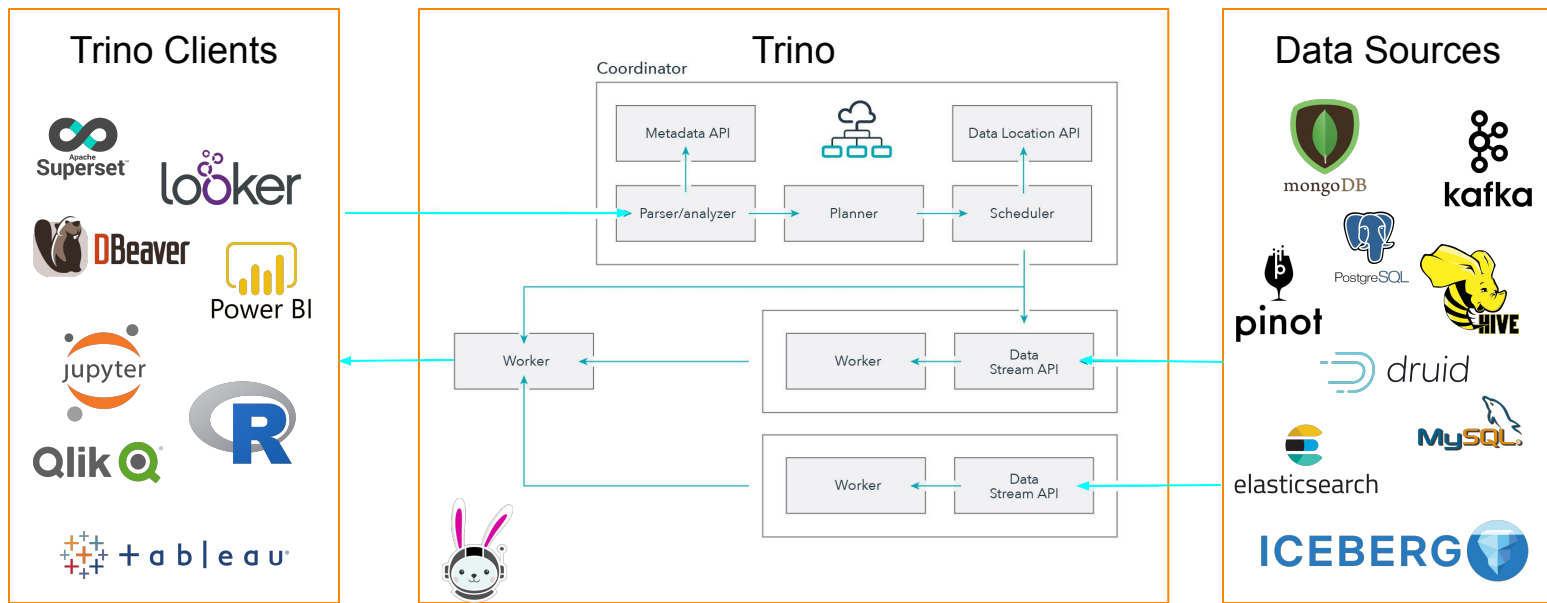
@bitsondatadev

# Overview

- Trino Overview
- Issues with Hive Table Format
- Iceberg Table Format
  - Column and Partition Model
  - Table Evolution
  - Cloud Compatibility
  - Concurrency Model
  - Time Travel
  - Iceberg Specification

# Trino TL;DL

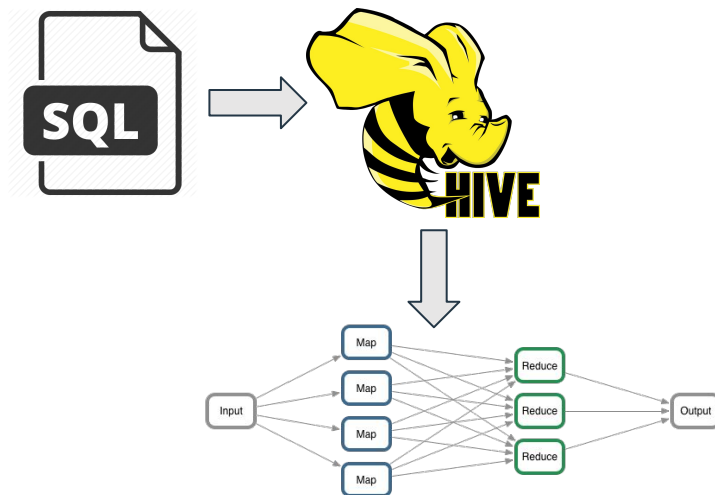
Trino is a fast distributed SQL query engine designed to query large data sets distributed over one or more heterogeneous data sources.



# Hive: The ~~perfect SQL~~ HiveQL solution

Developers at Facebook created Hive, a SQL-on-Hadoop solution that takes a SQL like syntax, HiveQL, and transforms it into MapReduce operations on data in Hadoop.

- Simplified development process
- Queries still taking long
- Established a base for how to model SQL tables of generic data stores



# Presto! Your queries are fast!

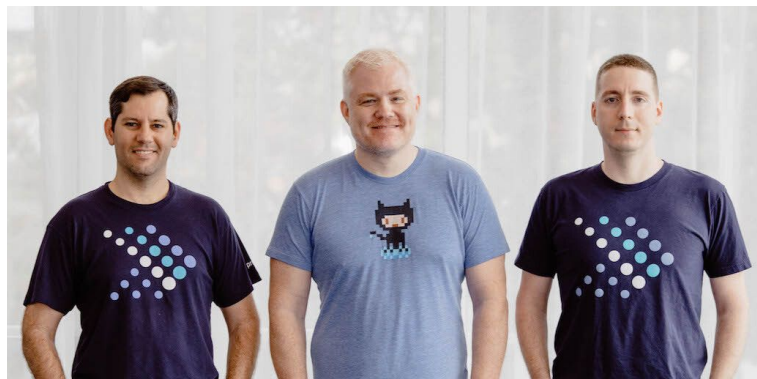


Martin Traverso, Dain Sundstrom, and David Phillips created Presto in 2012. It aimed to solve for the slow queries of Hive at Facebook and eventually many more.

## Development Philosophies:

- Open source with neutral governance model
- It just works (Netezza was a commercial inspiration)
- Fast, interactive analytics
- Correct results
- Standards based (ANSI SQL, JDBC, etc..)

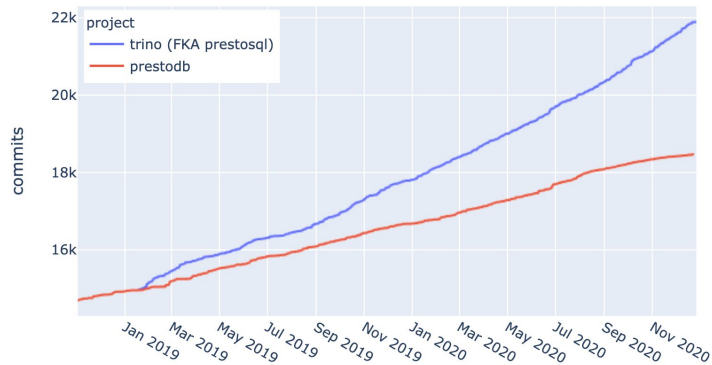
Facebook management unilaterally rewrites rules around committership in late 2018





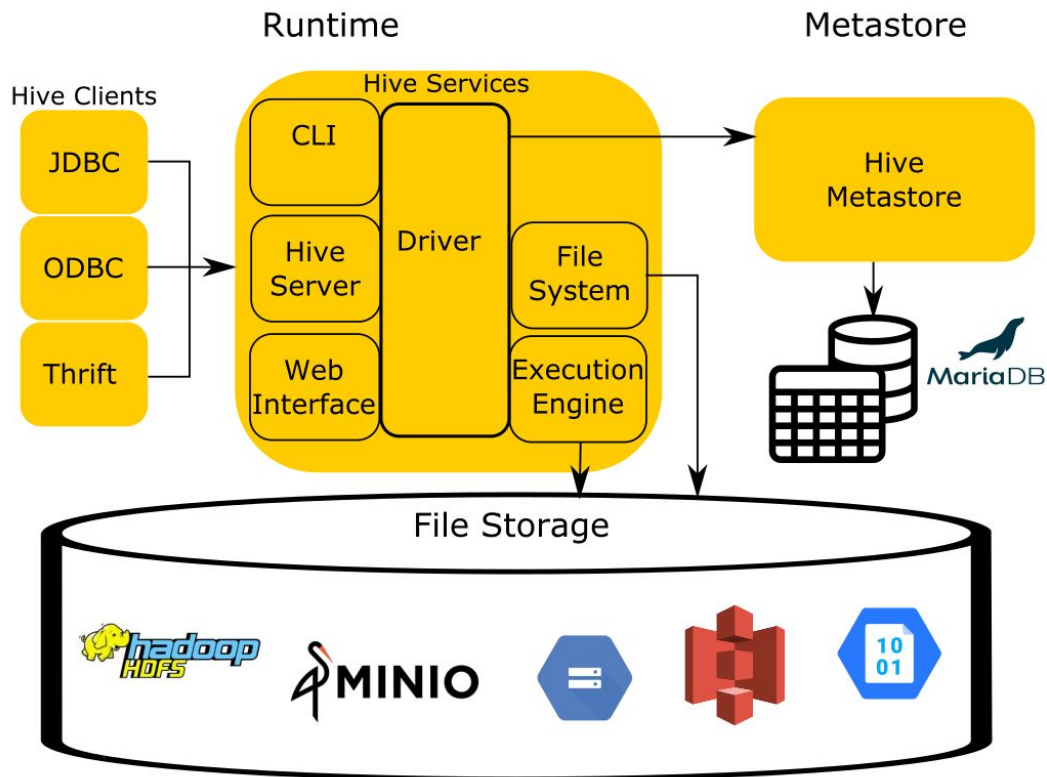
# Trino (formerly known as PrestoSQL)

- If you know Presto or are using PrestoSQL...
  - same software
  - same people
  - same community
  - under a shiny new name
  - and a cute bunny
- Companies don't run open source projects, people do.
- More details:

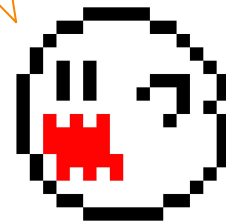


<https://trino.io/blog/2020/12/27/announcing-trino.html>

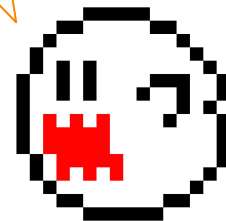
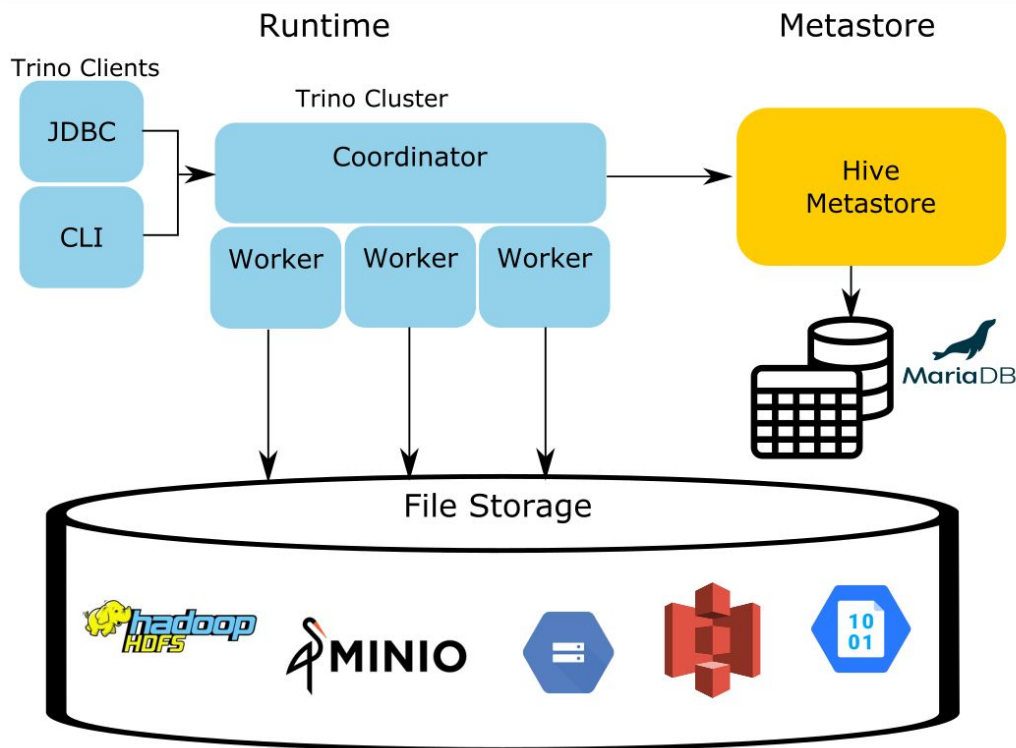
# Hive Architecture



I'm the Hive Specification Boooooo



# Trino Hive Connector Architecture





# Issues with Hive Table Format

- Invisible specification
- Column model
- Partition model
- Schema evolution requires data migration
- Not built for the cloud
- No transactional guarantees (depends on filesystem)
- Data and metadata not synchronized
- No concept of “time travel”
- No capability to roll data back

# Column and Partition Model

Create an events table partitioned on a TIMESTAMP field event\_time by day

Hive Statement

```
CREATE TABLE hive.logging.events (  
  level VARCHAR,  
  event_time TIMESTAMP,  
  message VARCHAR,  
  call_stack ARRAY(VARCHAR)  
) WITH (  
  format = 'ORC',  
  partitioned_by = ARRAY['event_time']  
);
```

Goal partition file org

```
events  
├─ event_time=2021-04-01  
│   ├─ event_time_2021-04-01_1.orc  
│   ├─ event_time_2021-04-01_2.orc  
│   └─ event_time_2021-04-01_3.orc  
│   └─ ...  
├─ event_time=2021-04-02  
│   └─ ...  
├─ event_time=2021-04-03  
│   └─ ...  
└─ ...
```

## Column and Partition Model

- Creation of Hive table in Trino
- Causes an exception due to partition location in statement

```
CREATE TABLE hive.logging.events (  
  level VARCHAR,  
  event_time TIMESTAMP,  
  message VARCHAR,  
  call_stack ARRAY(VARCHAR)  
) WITH (  
  format = 'ORC',  
  partitioned_by = ARRAY['event_time']  
);
```

Partition keys must be the last columns in the table and in the same order as the table properties: [event\_time]

## Column and Partition Model

- Move timestamp to the last column
- TIMESTAMP is at the second granularity rather than the day granularity

```
CREATE TABLE hive.logging.events (  
  level VARCHAR,  
  message VARCHAR,  
  call_stack ARRAY(VARCHAR),  
  event_time TIMESTAMP  
) WITH (  
  format = 'ORC',  
  partitioned_by = ARRAY['event_time']  
);
```

## Column and Partition Model

- A separate directory for every second you get an event file.
- Solution?

```
events
├── event_time=2021-04-01 00:00:00
│   └── event_time_2021-04-01-00:00:00_1.orc
├── event_time=2021-04-01 00:00:01
│   └── event_time_2021-04-01-00:00:01_2.orc
├── event_time=2021-04-01 00:00:03
│   └── event_time_2021-04-01-00:00:03_3.orc
└── ...
```

## Column and Partition Model

- Create duplicate VARCHAR that requires the awareness of the user
- Use this field

```
CREATE TABLE hive.logging.events (  
  level VARCHAR,  
  event_time TIMESTAMP,  
  message VARCHAR,  
  call_stack ARRAY(VARCHAR),  
  event_time_day VARCHAR  
) WITH (  
  format = 'ORC',  
  partitioned_by = ARRAY['event_time_day']  
);
```

# Column and Partition Model

- This works but...
- Inserts and reading require awareness of the second partition field
- Very error prone

```
events
├─ event_time_day=2021-04-01
│   ├─ event_time_day_2021-04-01_1.orc
│   ├─ event_time_day_2021-04-01_2.orc
│   ├─ event_time_day_2021-04-01_3.orc
│   └─ ...
├─ event_time_day=2021-04-02
│   └─ ...
├─ event_time_day=2021-04-03
│   └─ ...
└─ ...
```

# Column and Partition Model

- TIMESTAMP and the last partition field must be aligned
- No internal validation performed to verify these align

```
INSERT INTO hive.logging.events
VALUES
(
  'ERROR',
  timestamp '2021-04-01 12:00:00.000001',
  'Oh noes',
  ARRAY ['Exception in thread "main" java.lang.NullPointerException'],
  '2021-04-01'
),
(
  'ERROR',
  timestamp '2021-04-02 15:55:55.555555',
  'Double oh noes',
  ARRAY ['Exception in thread "main" java.lang.NullPointerException'],
  '2021-04-02'
),
(
  'WARN',
  timestamp '2021-04-02 00:00:11.1122222',
  'Maybe oh noes?',
  ARRAY ['Bad things could be happening??'],
  '2021-04-02'
);
```



## Column and Partition Model

- Must filter on both timestamp field, as well as partition field and be aware both exist
- Easy to build query incorrectly, especially with multiple partitions

```
SELECT *  
FROM hive.logging.events  
WHERE event_time < timestamp '2021-04-02';
```

```
SELECT *  
FROM hive.logging.events  
WHERE event_time < timestamp '2021-04-02'  
AND event_time_day < '2021-04-02';
```

```
|ERROR|2021-04-01 12:00:00|Oh noes|Exception in thread "main" java.lang.NullPointerException|
```

## Column and Partition Model

- All fixed in Iceberg using hidden partitioning
- Partitioning specification is abstracted from user

```
CREATE TABLE iceberg.logging.events (  
  level VARCHAR,  
  event_time TIMESTAMP(6),  
  message VARCHAR,  
  call_stack ARRAY(VARCHAR)  
) WITH (  
  partitioning = ARRAY['day(event_time)']  
);
```

# Column and Partition Model

- User only insert the timestamp field and mapping to day partitioning happens internally

```
INSERT INTO iceberg.logging.events
VALUES
(
  'ERROR',
  timestamp '2021-04-01 12:00:00.000001',
  'Oh noes',
  ARRAY ['Exception in thread "main" java.lang.NullPointerException']
),
(
  'ERROR',
  timestamp '2021-04-02 15:55:55.555555',
  'Double oh noes',
  ARRAY ['Exception in thread "main" java.lang.NullPointerException']],
(
  'WARN',
  timestamp '2021-04-02 00:00:11.112222',
  'Maybe oh noes?',
  ARRAY ['Bad things could be happening??']);
```

## Column and Partition Model

- Reader also doesn't need to be aware

```
SELECT *  
FROM iceberg.logging.events  
WHERE event_time < timestamp '2021-04-02';
```

```
|ERROR|2021-04-01 12:00:00|Oh noes|Exception in thread "main" java.lang.NullPointerException|
```

# Column and Partition Model

- Iceberg internally names the partition “event\_time\_day” based on transform function

```
events
├─ event_time_day=2021-04-01
│   ├─ event_time_day_2021-04-01_1.orc
│   ├─ event_time_day_2021-04-01_2.orc
│   ├─ event_time_day_2021-04-01_3.orc
│   └─ ...
├─ event_time_day=2021-04-02
│   └─ ...
├─ event_time_day=2021-04-03
│   └─ ...
└─ ...
```

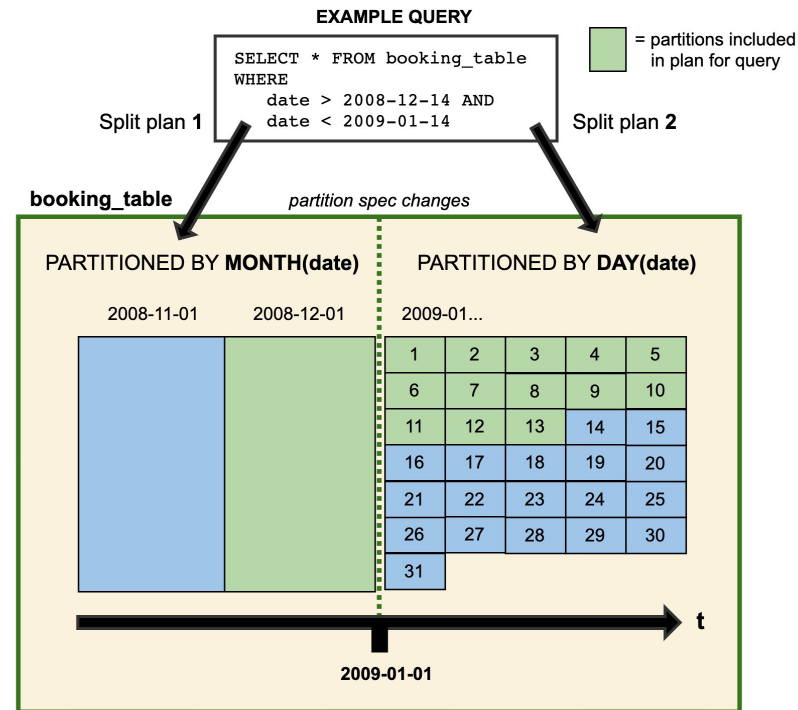
## Table Evolution: Partition Evolution

- Want to update the granularity of your partition?
- No support for in-place partition evolution in Hive
- Must perform a full table migration



# Table Evolution: Partition Evolution

- Iceberg supports in-place partition evolution
- Trino supports reads but not writes yet.
- Coming Soon!



# Table Evolution: Schema Evolution

- Limited support for in-place schema evolution in Hive
- May need to perform a full table migration

Hive 2.2.0 schema evolution based on filetype and operation.			
	Add	Delete	Rename
CSV/ TSV	✓	✗	✗
JSON	✓	✓	✗
ORC/ Parquet/ Avro	✓	✓	✗



## Table Evolution: Schema Evolution

- Iceberg only supports ORC, Parquet, and Avro to ensure guarantees

```
ALTER TABLE iceberg.logging.events  
ADD COLUMN severity INTEGER;
```

```
SELECT level, message, severity  
FROM iceberg.logging.events;
```

```
ERROR Double oh noes NULL  
WARN  Maybeh oh noes? NULL  
ERROR Oh noes          NULL
```

## Table Evolution: Schema Evolution

- Iceberg only supports ORC, Parquet, and Avro to ensure guarantees

```
INSERT INTO iceberg.logging.events VALUES
(
  'INFO',
  timestamp
  '2021-04-01 19:59:59.999999' AT TIME ZONE 'America/Los_Angeles',
  'es muy bueno',
  ARRAY ['It is all normal'],
  1
);
```

```
SELECT level, message, severity
FROM iceberg.logging.events;
```

ERROR	Double oh noes	NULL
WARN	Maybeh oh noes?	NULL
ERROR	Oh noes	NULL
INFO	es muy bueno	1

## Table Evolution: Schema Evolution

- Iceberg only supports ORC, Parquet, and Avro to ensure guarantees

```
ALTER TABLE iceberg.logging.events  
RENAME COLUMN severity TO priority;
```

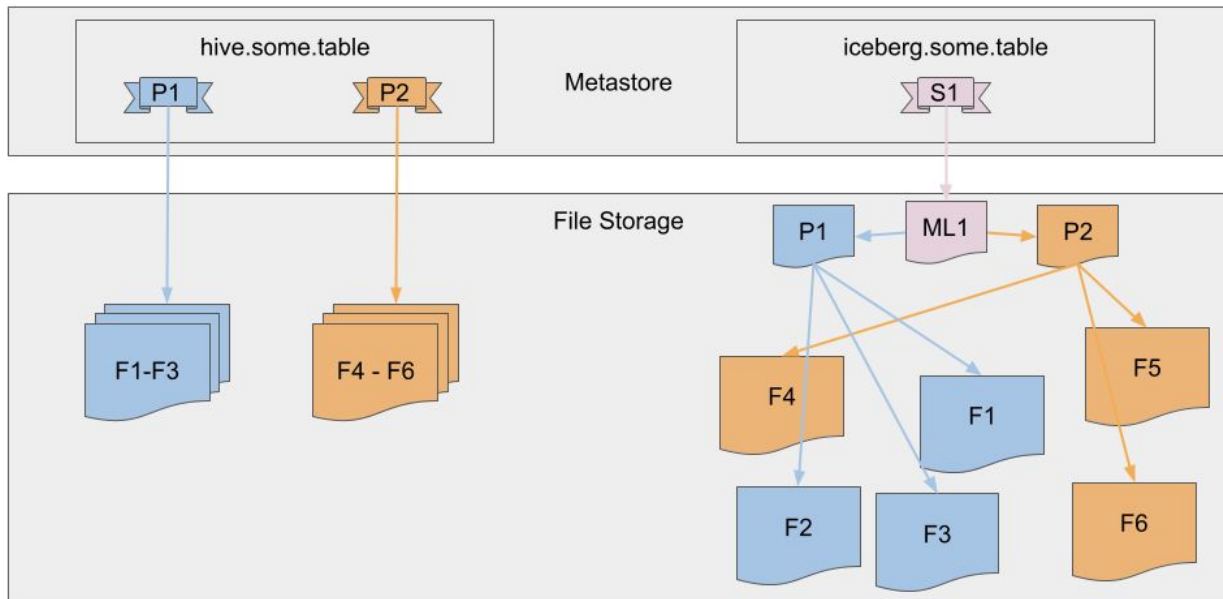
```
SELECT level, message, priority  
FROM iceberg.logging.events;
```

```
ERROR Double oh noes NULL  
WARN  Maybeh oh noes? NULL  
ERROR Oh noes          NULL  
INFO  es muy bueno     1
```

```
ALTER TABLE iceberg.logging.events  
DROP COLUMN priority;
```

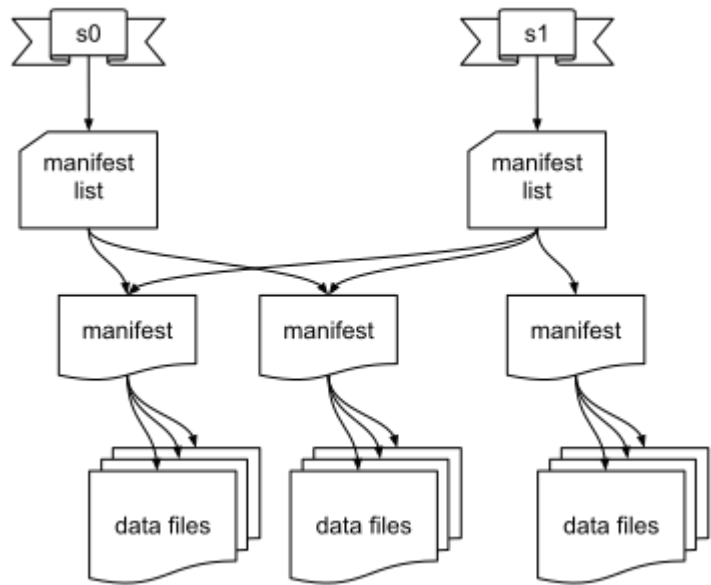
# Cloud Compatibility

- Avoid expensive list operation
- Track files not directories
- Store manifest files and accurate file statistics in persistent tree structure



# Concurrency Model

- Iceberg maintains a linear model of changes called immutable snapshots
- Optimistic concurrency between writers via lock-and-swap operation
- Serializable Isolation guarantees
- Git workflow analogy



# Time Travel

- Snapshots enable cool capabilities like time travel
- You can view snapshots of a table in Trino by appending '\$snapshots' to a table

```
SELECT level, message  
FROM iceberg.logging.events;
```

```
ERROR Double oh noes  
WARN  Maybeh oh noes?  
ERROR Oh noes
```

```
SELECT snapshot_id, parent_id, operation  
FROM iceberg.logging."events$snapshots";
```

snapshot_id	parent_id	operation
7620328658793169607		append
2115743741823353537	7620328658793169607	append

# Time Travel

- Snapshots are just pointers to manifest lists
- Manifest lists avro files containing a list of other manifest files (also avro)
- Manifest files contain the file stats and a list of data files

```
SELECT snapshot_id, parent_id, operation
FROM iceberg.logging."events$snapshots";
```

snapshot_id	parent_id	operation
7620328658793169607		append
2115743741823353537	7620328658793169607	append

```
SELECT manifest_list
FROM iceberg.logging."events$snapshots";
```

```
s3a://iceberg/logging.db/events/metadata/snap-7620328658793169607-1-cc857d8
9-1c07-4087-bdbc-2144a814dae2.avro
s3a://iceberg/logging.db/events/metadata/snap-2115743741823353537-1-4cb458b
e-7152-4e99-8db7-b2dda52c556c.avro
```

# Time Travel

- Each mutation to the data state creates a new snapshot

```
INSERT INTO iceberg.logging.events
VALUES
(
  'INFO',
  timestamp '2021-04-02 00:00:11.1122222',
  'It is all good',
  ARRAY ['Just updating you!']
);
```

```
SELECT snapshot_id, parent_id, operation
FROM iceberg.logging."events$snapshots";
```

snapshot_id	parent_id	operation
7620328658793169607		append
2115743741823353537	7620328658793169607	append
7030511368881343137	2115743741823353537	append

```
SELECT level, message
FROM iceberg.logging.events;
```

```
|ERROR|Oh noes      |
|INFO |It is all good |
|ERROR|Double oh noes |
|WARN |Maybeh oh noes?|
```



# Time Travel

- Append the '@<snapshot-id>' syntax to your table to time travel to a specific snapshot of your data

```
SELECT level, message  
FROM iceberg.logging."events@2115743741823353537";
```

```
|ERROR|Double oh noes |  
|WARN |Maybeh oh noes?|  
|ERROR|Oh noes      |
```

# Time Travel

```
CALL system.rollback_to_snapshot('logging', 'events', 2115743741823353537);
```

- Use the `system.rollback_to_snapshot(<schema>, <table>, <snapshot-id>)` function to permanently roll back to the given snapshot
- This doesn't erase the original snapshot

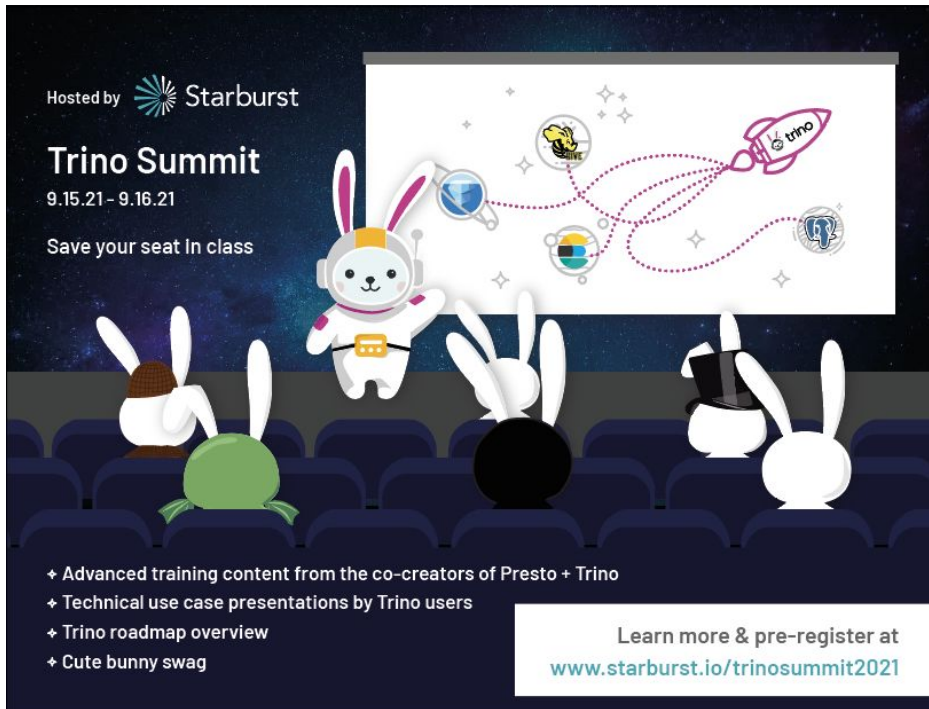
```
SELECT level, message  
FROM iceberg.logging.events;
```

```
|ERROR|Double oh noes |  
|WARN |Maybeh oh noes?|  
|ERROR|Oh noes      |
```

# Iceberg Specification

- Iceberg Spec: <https://iceberg.apache.org/spec/>
- No specific implementation of Iceberg is required, even the current Iceberg library.
- Any query engine can implement the spec as a standard to be compliant.
- Community is primarily focused on the specification over implementation.

# Community

A promotional graphic for the Trino Summit 2021. It features a dark blue background with a starry space theme. In the foreground, several stylized bunnies are seated in an audience, facing a large screen. The screen displays a diagram with a rocket labeled 'trino' and various icons representing different technologies and communities. The text on the left side of the graphic includes 'Hosted by Starburst', 'Trino Summit', '9.15.21 - 9.16.21', and 'Save your seat in class'. At the bottom left, there is a list of topics: 'Advanced training content from the co-creators of Presto + Trino', 'Technical use case presentations by Trino users', 'Trino roadmap overview', and 'Cute bunny swag'. At the bottom right, there is a white box with the text 'Learn more & pre-register at www.starburst.io/trinosummit2021'.

- Trino Slack
  - <https://trino.io/slack.html>
- Trino Community Broadcast
  - <https://trino.io/broadcast/>
- Trino Virtual Meetups
  - <https://www.meetup.com/trino-america>
  - <https://www.meetup.com/trino-emea>
  - <https://www.meetup.com/trino-apac>
- Contribute to the project
  - <https://trino.io/development/>
- Write blogs or docs
  - <https://trino.io/blog/>
  - <https://github.com/trinodb/trino/tree/master/docs>



# Thank you

Please give us a ★ on  
[github.com/trinodb/trino](https://github.com/trinodb/trino)

 @bitsondatadev