Apache Iceberg ingestion with Apache NiFi

Lester Martin; Trino Developer Advocate @ Starburst

Berlin Buzzwords 2025

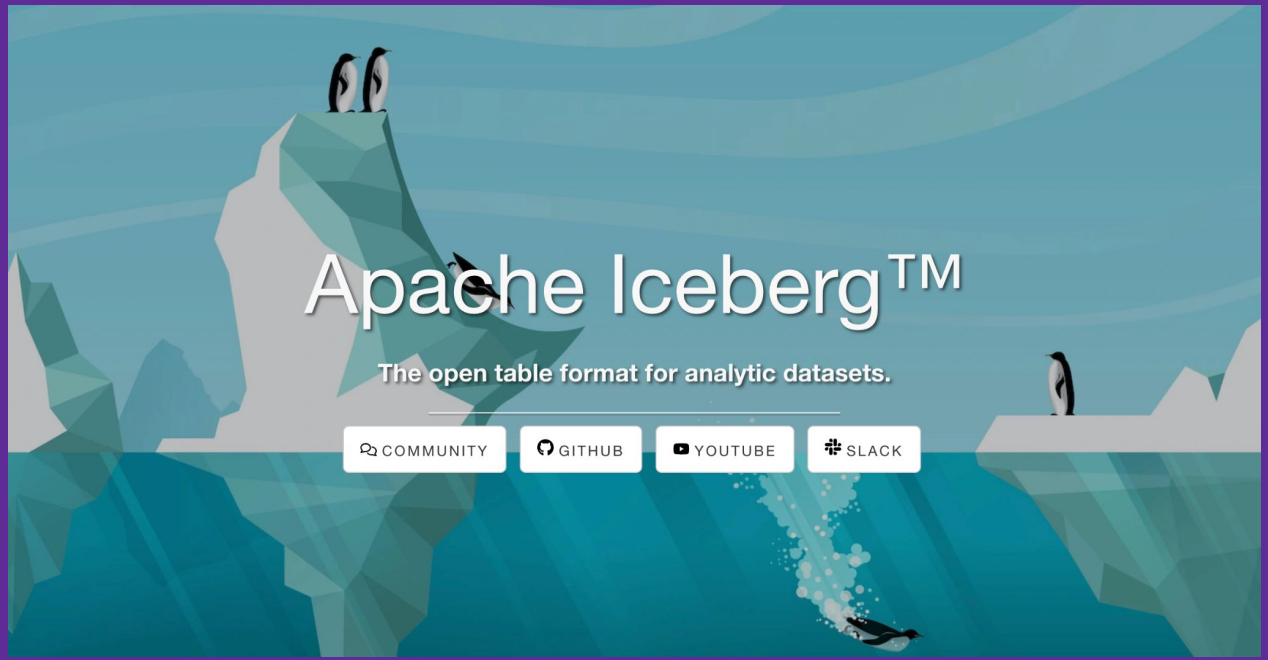
Connection before content

Lester Martin – https://linktr.ee/lestermartin

- Trino developer advocate @ Starburst
- 30+ years of technology experience
  - Started my journey on a TRS-80 Model III
  - Played most every role, but a programmer at my core
  - ½ career in OLTP and ½ in data analytics
  - Decade+ of “big data” experience to include
    - Trino/Starburst, Hadoop, Hive, Spark
    - NiFi, Kafka, Storm, Flink
    - HBase, MongoDB

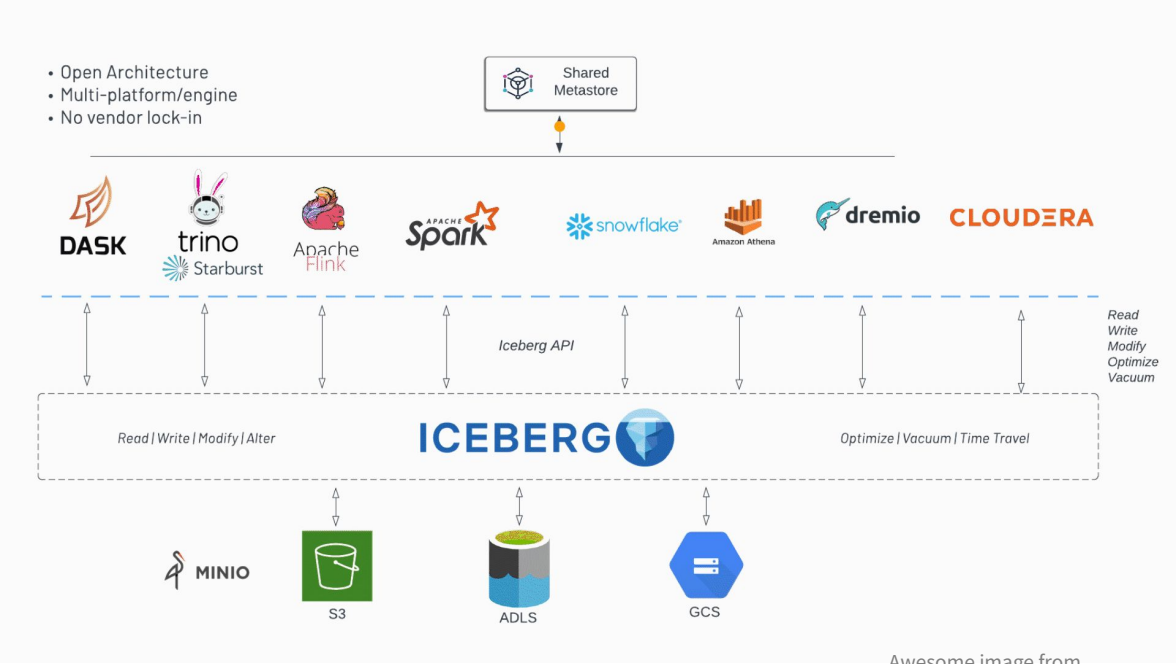
Apache Iceberg

What is it?



Multi-engine support for ludicrously large tables

Iceberg is a **high-performance** format **for huge analytic tables**. Iceberg brings reliability and simplicity of SQL tables to big data, while making it possible **for engines** like Spark, Trino, Presto, Hive and Impala **to safely work with the same tables, at the same time**.



Awesome image from
https://medium.com/@karim.faiz/what-is-apache-iceberg-184c586ff7f0

ACID-compliant transactions

`INSERT`, `UPDATE`, `DELETE`, **and `MERGE`** statements

Only single-statement boundaries (aka auto-commit) *today*

Supports structural evolution

Schema evolution changes are independent and free of side-effects, without rewriting files

Iceberg even supports partition evolution!!

```
CREATE TABLE orders_iceberg
  WITH (partitioning =
    ARRAY['month(orderdate)']) AS
SELECT * FROM tpch.sf1.orders;

ALTER TABLE orders_iceberg
  SET PROPERTIES partitioning =
    ARRAY['day(orderdate)']);
```

- orderdate_day=1996-06-13/
- orderdate_day=1996-09-26/
- orderdate_day=1997-11-04/
- orderdate_month=1992-01/
- orderdate_month=1992-02/
- orderdate_month=1992-03/
- orderdate_month=1992-04/
- orderdate_month=1992-05/

Table versioning via snapshots

Content and structural changes create new snapshots which enable

- Point in time reads (time-travel)
- Branching & tagging
- Rollbacks

Architecture overview

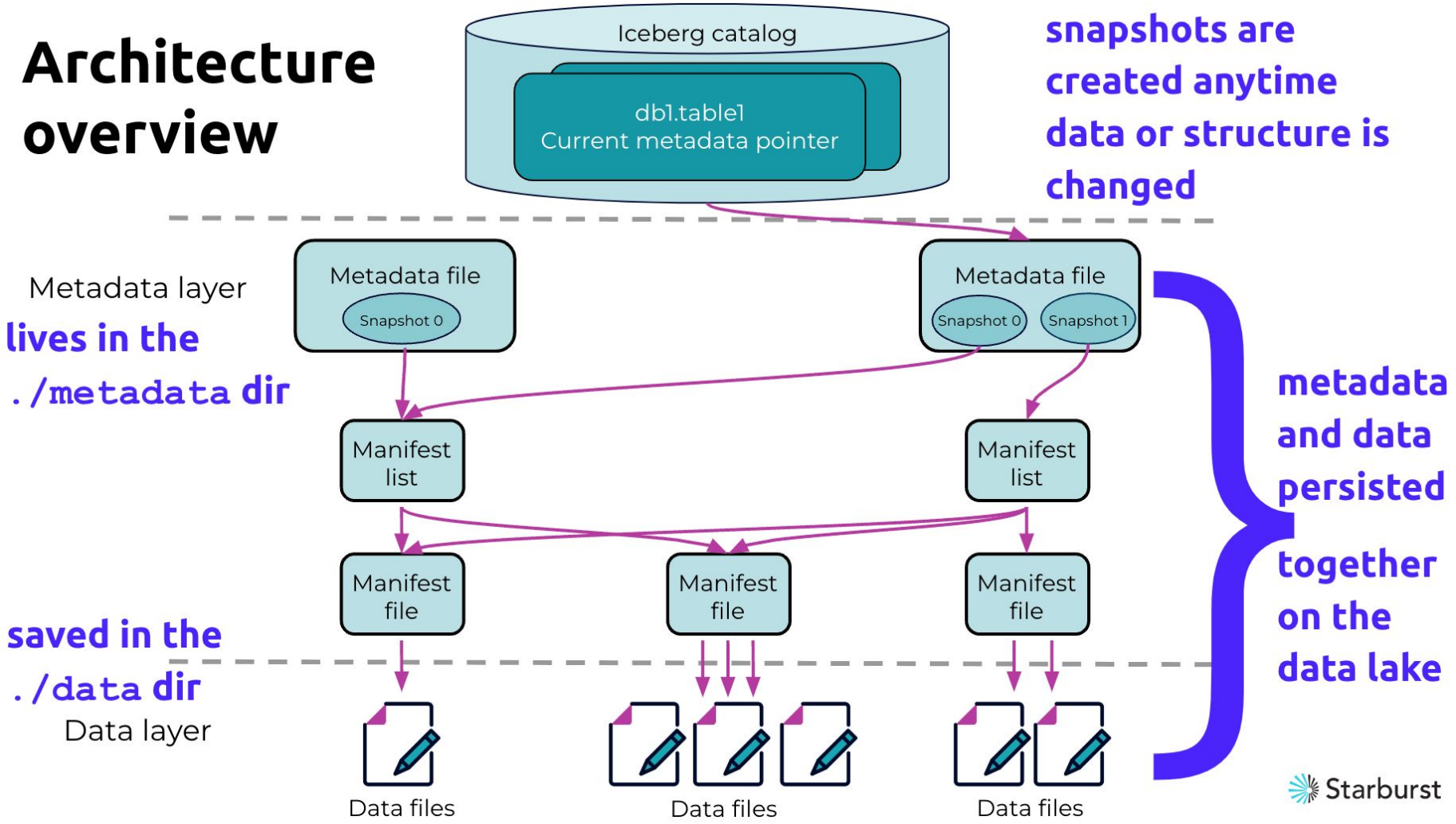
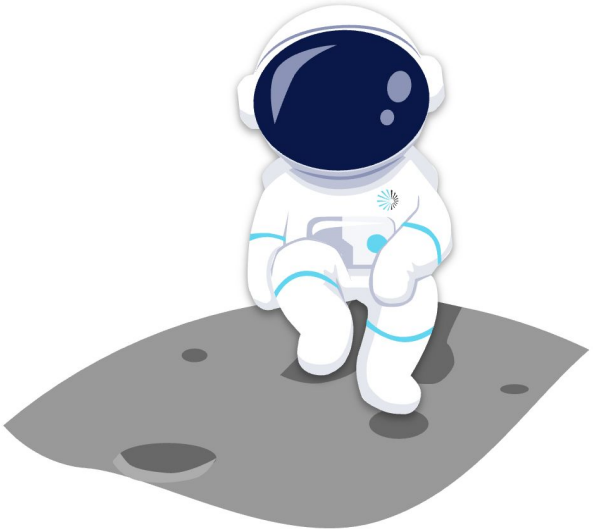


Table maintenance is required

Immutable files coupled with metadata sprawl are kept in check with cleanup functions

- Compaction
- Recalculate statistics
- Snapshot expiration
- Delete orphaned files



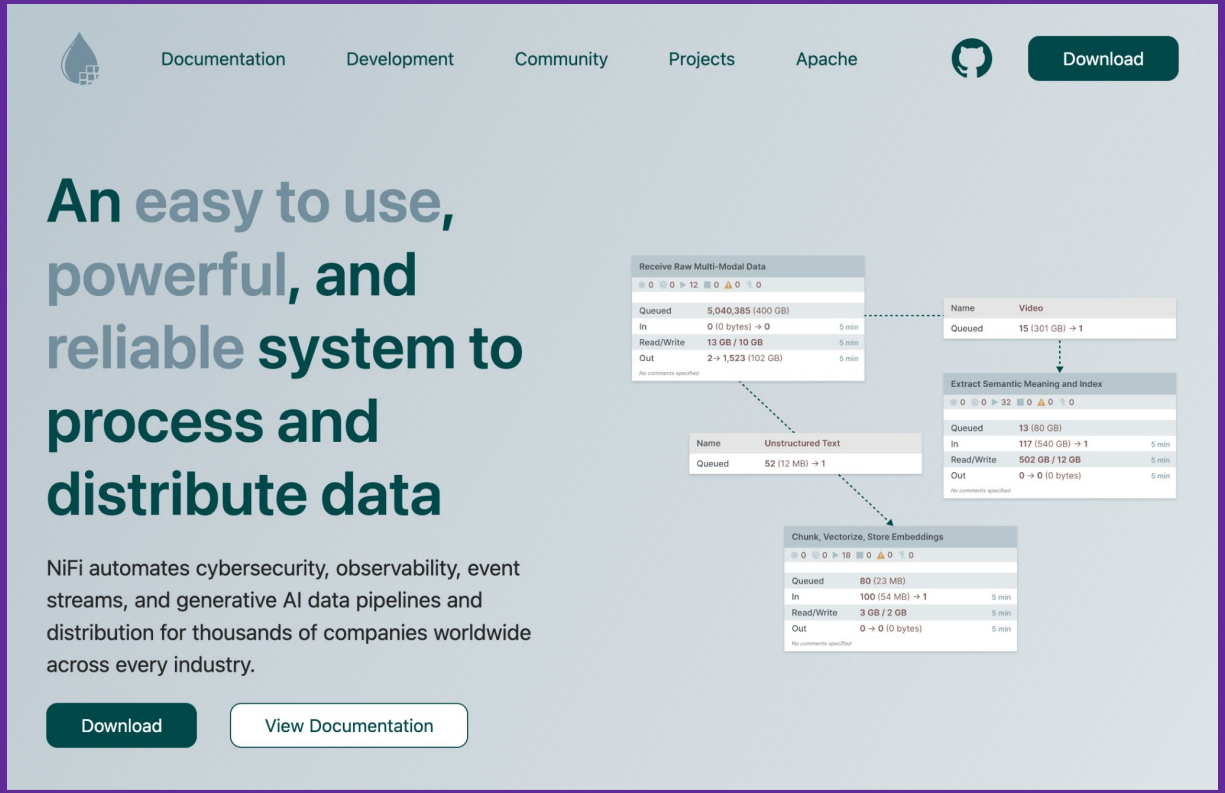
data maintenance jobs

Demo

Create, populate & query a table

Apache NiFi

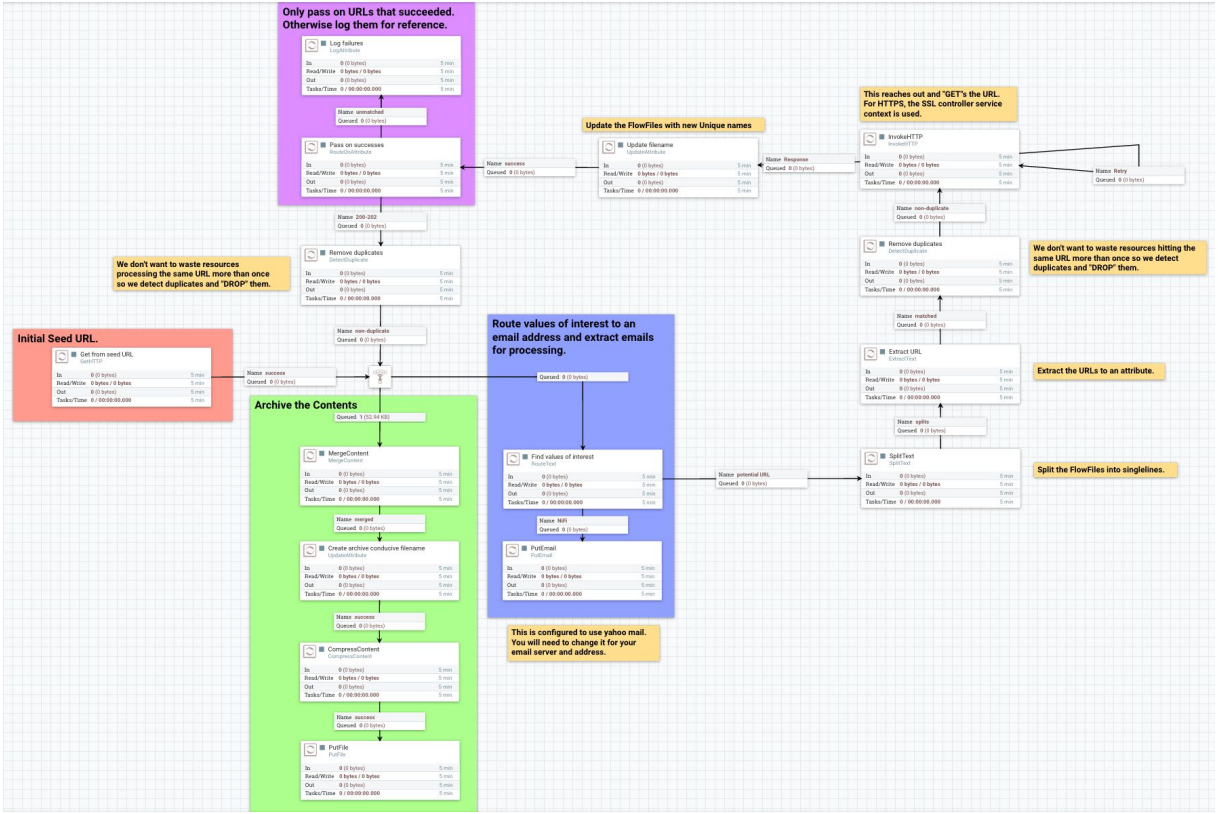
What is it?



Low/no-code dataflow (pipeline) creation tool

NiFi was built to **automate the flow of data between systems**.

While the term **‘dataflow’** is used in a variety of contexts, we use it here to mean the **automated and managed flow of information between systems**.



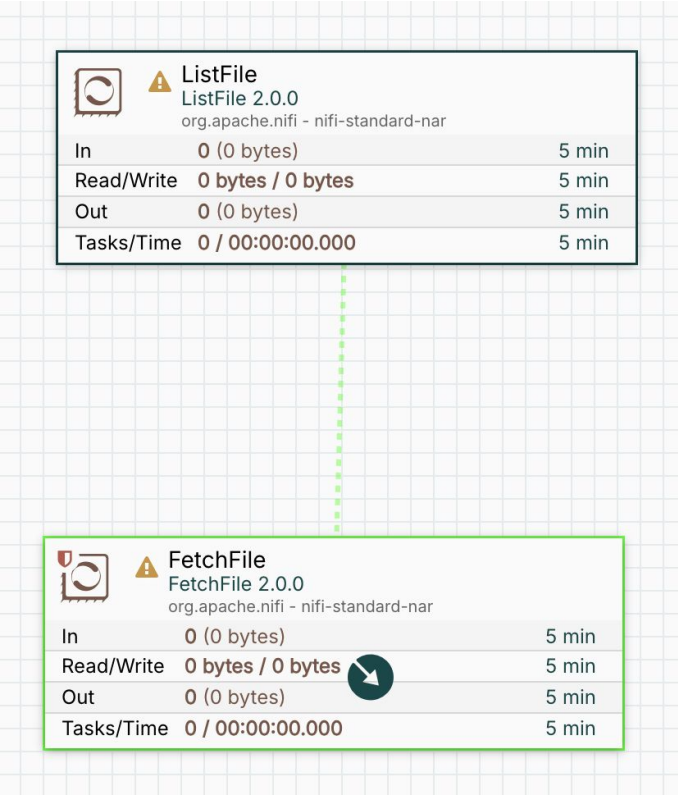
Check out my NiFi intro tutorial at https://learn-with-lester.freesite.online/tutorials/flow-dev-intro/

Batch or streaming?

YES! Both!!

and many more connectors!

Development approach



Drag processors from the toolbar to the canvas

- Data ingest
- Transformations, modifications & conversions
- Activities such as table lookups
- Routing & mediation
- Data egress

Create connections between the processors

- Backed by queues
- Backpressure controls
- Configurable prioritization

NiFi tackles the full development life cycle

The NiFi UI also **addresses the rest of the development life cycle**

NiFi tackles the full development life cycle

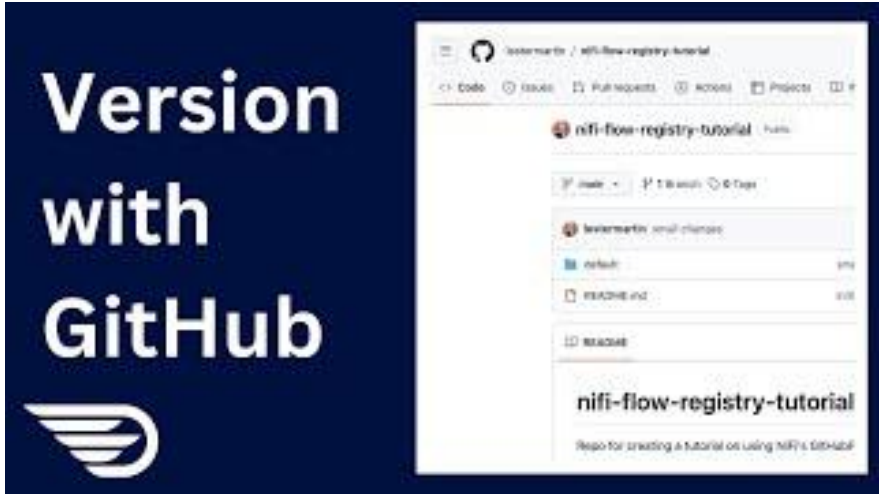
The NiFi UI also **addresses the rest of the development life cycle**

- **Compile & build** – No such thing as the dataflow is ready to run
    - Custom processors can be created with Python and Java

NiFi tackles the full development life cycle

The NiFi UI also **addresses the rest of the development life cycle**

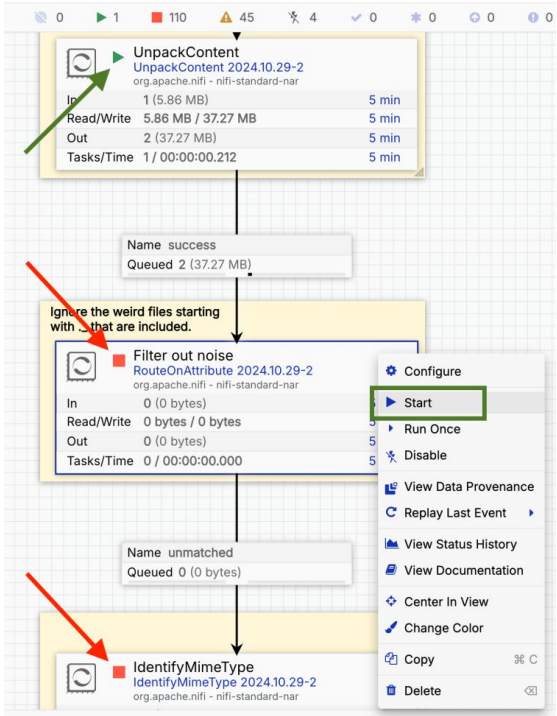
- **Compile & build** – No such thing as the dataflow is ready to run
  - Custom processors can be created with Python and Java
- **Deploy** – Just as above, nothing to do
  - Versioning and dev/test/prod movements possible *& recommended*



NiFi tackles the full development life cycle

The NiFi UI also **addresses the rest of the development life cycle**

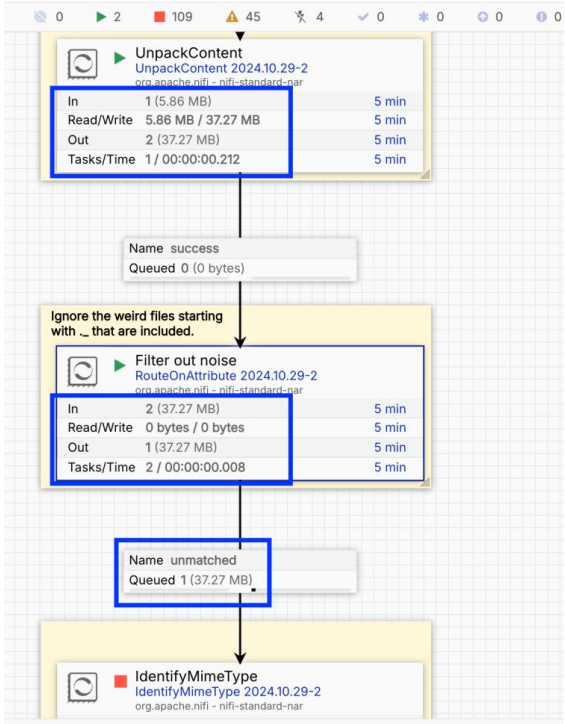
- **Compile & build** – No such thing as the dataflow is ready to run
  - Custom processors can be created with Python and Java
- **Deploy** – Just as above, nothing to do
  - Versioning and dev/test/prod movements possible *& recommended*
- **Runtime** – NiFi is the execution environment
  - Each processor is independently started/stopped
  - Can scale from a single node to 1000+ node clusters



NiFi tackles the full development life cycle

The NiFi UI also **addresses the rest of the development life cycle**

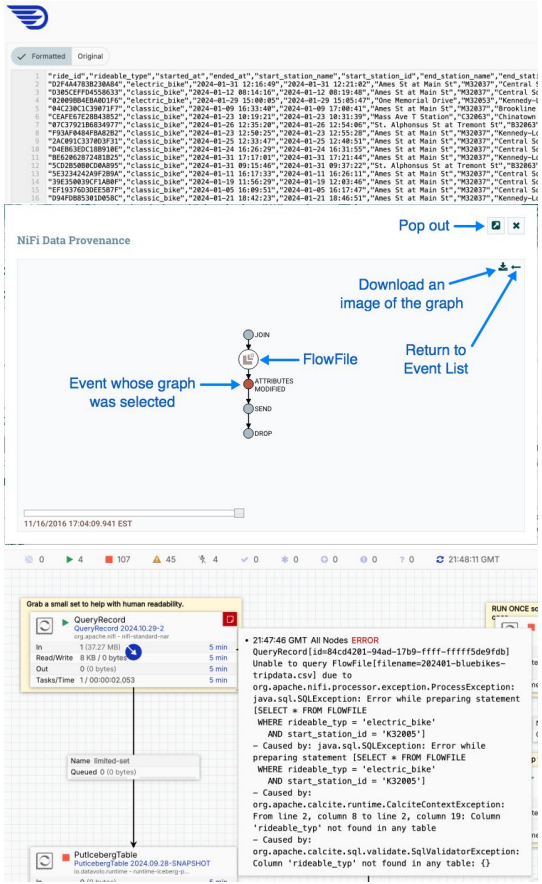
- **Compile & build** – No such thing as the dataflow is ready to run
  - Custom processors can be created with Python and Java
- **Deploy** – Just as above, nothing to do
  - Versioning and dev/test/prod movements possible *& recommended*
- **Runtime** – NiFi is the execution environment
  - Each processor is independently started/stopped
  - Can scale from a single node to 1000+ node clusters
- **Monitoring & reporting** – Processors & connection queues provide detailed statistics



NiFi tackles the full development life cycle

The NiFi UI also **addresses the rest of the development life cycle**

- **Compile & build** – No such thing as the dataflow is ready to run
  - Custom processors can be created with Python and Java
- **Deploy** – Just as above, nothing to do
  - Versioning and dev/test/prod movements possible *& recommended*
- **Runtime** – NiFi is the execution environment
  - Each processor is independently started/stopped
  - Can scale from a single node to 1000+ node clusters
- **Monitoring & reporting** – Processors & connection queues provide detailed statistics
- **Tracing & debugging** – Comprehensive data viewing & lineage exists as well as log viewing exists



NiFi tackles the full development life cycle

The NiFi UI also **addresses the rest of the development life cycle**

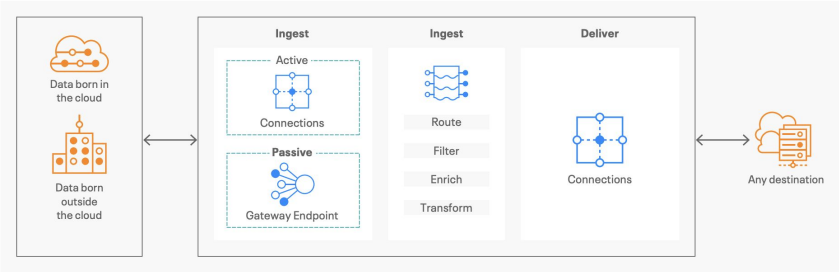
- **Compile & build** – No such thing as
  - Custom processors can be crea
- **Deploy** – Just as above, nothing to d
  - Versioning and dev/test/prod r
    recommended
- **Runtime** – NiFi is the execution envi
  - Each processor is independent
  - Can scale from a single node to
- **Monitoring & reporting** – Processo
  provide detailed statistics
- **Tracing & debugging** – Comprehen
  exists as well as log viewing exists

Demo

Explore a simple dataflow

Cloudera & Snowflake OSS+ products

CLOUDERA DATAFLOW

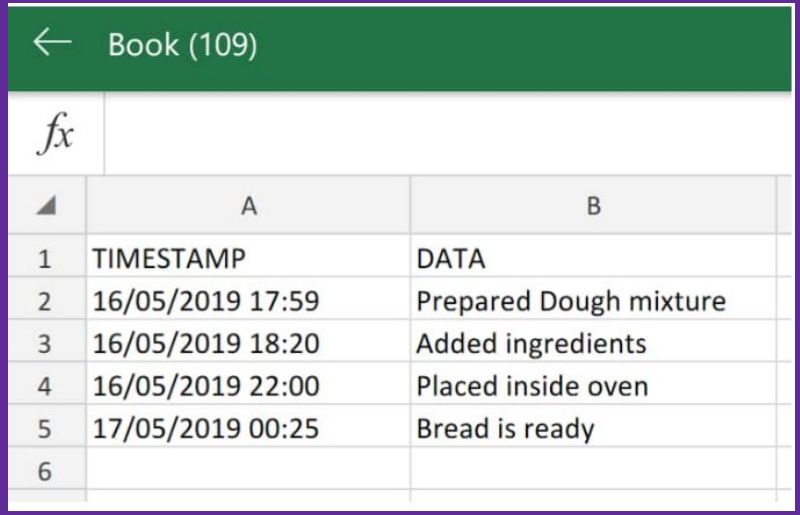


powered by

APACHE nifi

Ingestion strategies

Timestamped immutable data



Isn’t there a PutIceberg processor?

Well, yes and no, but first here is what it does

- Bulks up a bunch of records or files coming into it
- Converts them into files using the Iceberg table’s file format (ex: Apache Parquet)
- Moves those files to the table’s location on the data lake
- Uses the Iceberg API to include the new records from the file(s) created (creates a new snapshot)

Sounds like a good thing, but…

Status of the PutIceberg processor

Cloudera Dataflow

Really only works with Cloudera Data Platform (i.e. Hadoop) which does allow object store persistence

Snowflake Openflow

A replacement processor, PutIcebergTable, was created, but only works with the Apache Polaris metadata catalog

Status of the PutIceberg processor

Cloudera Dataflow

Really only works with Cloudera Data Platform (i.e. Hadoop) which does allow object store persistence

Not OSS

Snowflake Openflow

A replacement processor, PutIcebergTable, was created, but only works with the Apache Polaris metadata catalog

Not OSS

Status of the PutIceberg processor

Cloudera Dataflow

Really only works with Cloudera Data Platform (i.e. Hadoop) which does allow object store persistence

Not OSS

Snowflake Openflow

A replacement processor, PutIcebergTable, was created, but only works with the Apache Polaris metadata catalog

Not OSS

Pure OSS Apache NiFi approach

Essentially, we just need to replicate the steps listed on prior slide

Strategically, “someone” needs to enhance PutIceberg to work with HDFS **and** popular object stores & a variety of catalog/metastores (not just HMS)

Demo

Emulate PutIceberg behavior

Caveats of the PutIceberg strategy

Seems about perfect for append-only use cases, but there are concerns

- Data engineer needs to know which file format is used
- Difficult task to make sure the files meet other table configurations
  - Partitioning
  - Sorting
  - Bucketing
  - Bloom filters
- File type conversions happen on NiFi cluster instead of analytics compute engine
- Files can be deleted over time as part of table maintenance

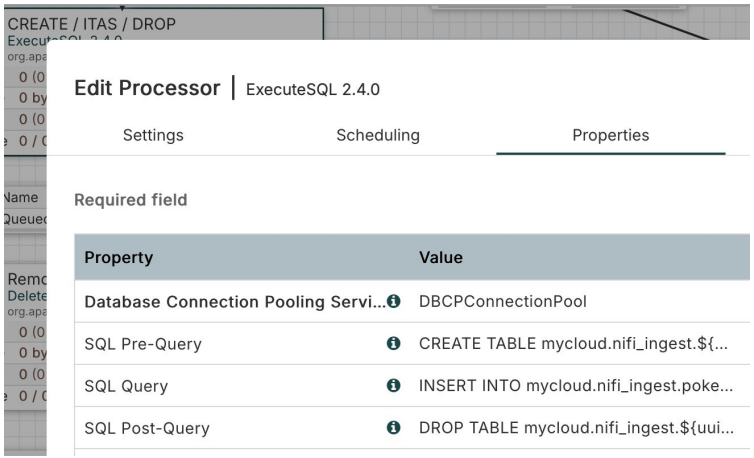
Okay… is there an alternative?

Ephemeral external table strategy

Build a highly reusable dataflow that

- Bulks up a bunch of records or files coming into it – *leave it in whatever format it arrives in*
- Save files to data lake
- Create a Hive external table
- Perform INSERT INTO (or MERGE) SQL statement on the compute engine – *let the analytics cluster handle the file conversion and the any advanced table configurations*
- Drop ephemeral table
- Move or delete the data lake files

Just as easy as before!

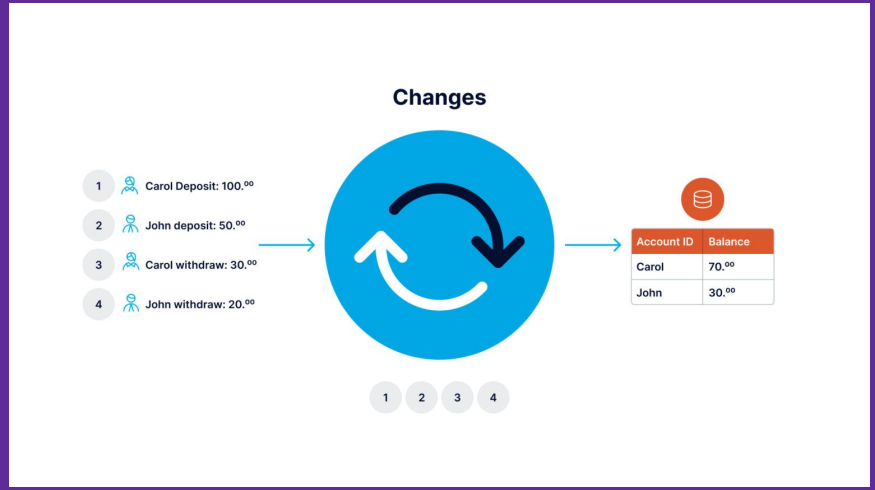


Demo

Use the ephemeral table strategy

Ingestion strategies

Table synchronization (CDC)



KISS principle can be appropriate

Keep it simple if

- The data really isn’t BIG data
- You can wait a period of time for the sync-up
- You don’t need to replay every change, you just need it to be accurate as-of refresh

KISS principle can be appropriate

Keep it simple if

- The data really isn't BIG data
- You can wait a period of time for the sync-up
- You don't need to replay every change, you just need it to be accurate as-of refresh

Build a simple pipeline that

- Builds a new version of the table
- Drop (or rename) the active table
- Rename the replacement table to be the active table
- *Rinse, lather, repeat*

Otherwise… just use the ephemeral table strategy

Easy peasy…

Just swap out the INSERT statement with an appropriate MERGE statement!

Yabba Dabba Do!

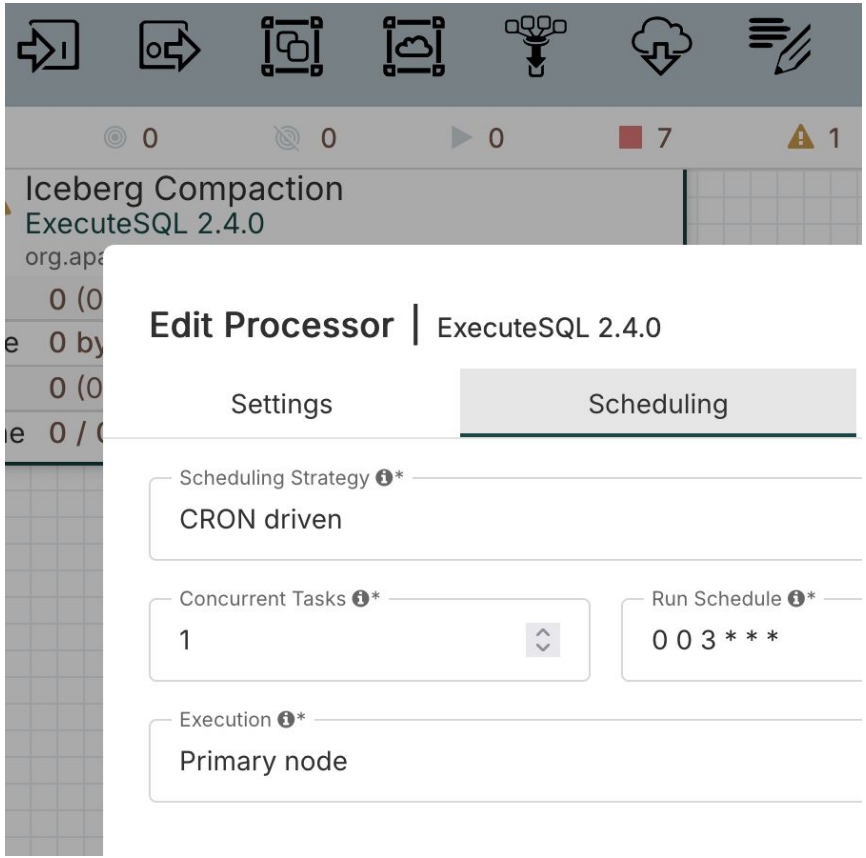
Table maintenance

No problem...

Let NiFi be your scheduler and invoker

Processors can be run on a schedule which is perfect for

- Compaction
- Recalculate statistics
- Snapshot expiration
- Delete orphaned files



Thank you!

Lester Martin

https://linktr.ee/lestermartin

Trino Developer Advocate @ Starburst

https://devcenter.starburst.io/

Scan for a Trino and Iceberg cheat sheet

Source files - https://github.com/lestermartin/events/tree/main/2025-06-16_NiFi2Iceberg