Preventative Metadata: Building for Data Reliability with DataHub, GE, & Airflow

John Joyce | Co-Founder | Acryl Data
Tamas Nemeth | Software Engineer | Acryl Data

Airflow Summit 2022
About Us

John Joyce
Co-Founder / Engineer

Tamas Nemeth
Software Engineer
About Acryl Data

Company
Founded early 2021 by data engineers from LinkedIn, Airbnb

What we do
Bring clarity & control to complex data ecosystems by driving forward the open source DataHub project

Learn more at acryldata.io
Agenda

1. What is DataHub?
2. What is Data Reliability?
3. Building for Data Reliability
What is DataHub?
What is DataHub?

DataHub is an open source metadata platform that enables Data Discovery, Data Observability, and Federated Governance on top of a high-fidelity Metadata Graph.

Learn more [datahubproject.io](http://datahubproject.io)
What is DataHub?

See it in action! → demo.datahubproject.io
What is DataHub?

The #1 Open Source Metadata Platform

Integrations

Adopters

Acryl Data
Community

3,193 Slack Members
10x YoY Growth
Across 56 Countries & 27 Local Time Zones

Top Member Roles
- Data Engineer
- Software Engineer
- System Architect
- Data Team Lead
- Eng Manager
- Product Manager
- Data Scientist

Top Member Industries
- Software
- Ecommerce
- Info Tech
- FinTech

5.5k GitHub Stars
801 YouTube Subscribers
397 Blog Subscribers
The DataHub Way

MetOps Principles

Metadata 360

Bridge the gap between technical and logical metadata to create a “360-view”

Shift Left

Declare metadata at source
Collect metadata in real time

Active Metadata

Put metadata to work in the operational plane
What is Data Reliability?
What is Data Reliability?

Reliable → “consistently good in quality or performance. Able to be trusted.” - Oxford dictionary

Reliability → “the overall consistency of a measure” - Wikipedia

Data Reliability can be thought of as the overall consistency of ___________

Data Quality
Quality vs. Reliability

Data Quality

- Correctness
- Timeliness
- Availability

Data Reliability

Less Reliable

Quality vs. Time

More Reliable

Quality vs. Time

Acryl Data
Realizing Data Reliability

Correctness

Timeliness / Completeness

Availability

At Scale

Time
Why should I care?

Data is becoming a **product**.

*Availability, timeliness, correctness will continue to grow in importance*
Challenges

Scale

Complexity

...And changing quickly!
Challenges

An emergent challenge: Separating **signal** from **noise**
Building for Data Reliability
Independent producer and consumers
Pattern 1: Consumer-side Validation
Pattern 1: Consumer-side Validation

Downsides: ad-hoc / inconsistent, partial coverage, duplicative efforts
Pattern 2: Async Validation

**Producer**

- Transform
- Table
- Validations

**Run periodically**

**Consumers**

- Transform
  - Table
- Transform
  - Chart
- Transform
  - Emails

Acryl Data
Pattern 2: Async Validation

- Id column is distinct
- Age column is not null
- Row count > 1000 and < 2000
- Stdev of height column < 6.5

Transform → Table → Validations

Transform → Table
Transform → Chart
Transform → Emails

Run periodically
Pattern 2: Async Validation

Contract-*after-write*

- Transform
- Table
- Validations
- Alerts

- Id column is distinct
- Age column is not null
- Row count > 1000 and < 2000
- Stdev of height column < 6.5

Consumers

- Transform
- Table
- Chart
- Emails

Run periodically

Downside: Bad data propagates by default
Can we do better?
An improvement: Sync Validation

The Communication Problem: How do consumers know?

**Contract-on-write**

**Upsides:** Consistency, coverage, centralization
A New Approach

**Metadata-Driven Orchestration**

[Diagram of orchestration process with nodes labeled Producer Dag, Transform, Table, Validation, Reporting, and Consumers (Table, Chart, Emails)].

Acryl Data
A New Approach

1. Report metadata
A New Approach

2. Check metadata

"Inserted 45 rows into Table"
"Table passed validation suite"

"Is input data up to date?"
"Is it passing validations?"
A New Approach

2. Check metadata

1. Report

"Inserted 45 rows into Table"
"Table passed validation suite"

"Is input data up to date?"
"Is it passing validations?"

Producer

Producer Dag

Transform

Table

Validation

Reporting

Metadata Platform

Run

Check

Transform

Table

Consumers

Check

Transform

Chart

Check

Transform

Emails
A New Approach

2. Check metadata

"Inserted 45 rows into Table"
"Table passed validation suite"

"Is input data up to date?"
"Is it passing validations?"
A Practical Example
A pipeline from scratch

Prospective Adopters
A problem: Delayed Data

One day...

Prospective Adopters
A problem: Delayed Data

One day...

Pet Profiles -> Snowflake loader -> Pet Profiles -> Email Sender

Ecommerce Airflow -> Marketing Airflow

Delayed!  Delayed!  Still runs

...bad emails

Prospective Adopters

Acryl Data
DataHub Operations

Step 1: Report

"INSERTed 1000 new rows into Pet Profiles on 5-22-2022 at 10am PST"
Reporting Operations

```python
def report_operation(context):
    hook = DatahubRestHook("datahub_longtail")
    host, password, timeout_sec = hook._get_config()
    reporter = OperationReporter(
        datahub_host=host, datahub_token=password, timeout=timeout_sec
    )
    task = context["ti"].task
    for outlet in task.outlets:
        print(f"Reporting insert operation for {outlet.urn}")
        reporter.report_operation(urn=outlet.urn, operation_type="INSERT")

pet_profiles_load = BashOperator(
    task_id="load_s3_adoption_pet_profiles",
    dag=dag,
    inlets=[Dataset("s3", "longtail-core-data/mongo/adoption/pet_profiles")],
    outlets=[Dataset("snowflake", "long_tail_companions.adoption.pet_profiles")],
    bash_command="echo Dummy Task",
    on_success_callback=report_operation,
)
```

- Define Inlets and outlets with Datahub Dataset
- Report operation data on success
- Setup a datahub connection
- Create an operation reporter
- Report operation data for all task outlets to Datahub
DataHub Operations

Step 2: Verify

"INSERTed 1000 new rows into Pet Profiles on 5-22-2022 at 10am PST"

Bonus: Report custom metadata!

"Have new rows been INSERTed in the past 12 hours?"

Bonus: Verify custom metadata!

- Ecommerce Airflow
  - Pet Profiles
  - Snowflake loader
  - Pet Profiles
  - Email Sender

- Marketing Airflow
DataHub Operations Circuit Breaker

```python
pet_profiles_operation_sensor = DatahubOperationCircuitBreakerSensor(
    task_id="pet_profiles_operation_sensor",
    datahub_rest_conn_id="datahub_longtail",
    urn=[
        "urn:li:dataset:
        (urn:li:dataPlatform:snowflake, long_tail_companions.adoption.pet_profiles, PROD)
    ],
    time_delta=datetime.timedelta(hours=12),
)
```

- Set up an Operation Circuit Breaker Sensor
- List of dataset urns to check for operation data
- The time delta we expect to have operational data

*Also available as operator, and direct Python API*
Another problem: Broken Data

A few months later...

timestamp column changed from seconds to milliseconds
Another problem: Broken Data

A few months later...

t > x && t < y
Another problem: Broken Data

A few months later...

t > x && t < y

...no emails
DataHub Assertions

Step 1: Validate + Report

- Id column is distinct
- Age column is not null
- Row count > 1000 and < 2000
- Timestamp col in millis

Ecommerce Airflow

Validation

Marketing Airflow

Pet Profiles

Snowflake loader

Pet Profiles

Email Sender
Reporting Assertions

Step 1: Define Assertions

```json
{
    "expectation_type": "expect_table_row_count_to_be_between",
    "kwargs": {
        "min_value": 48000,
        "max_value": 50000
    },
    "meta": {}
},
{
    "expectation_type": "expect_column_values_to_be_in_set",
    "kwargs": {
        "column": "sex",
        "value_set": ["M", "F"]
    },
    "meta": {}
},
{
    "expectation_type": "expect_column_values_to_not_be_null",
    "kwargs": {
        "column": "sex"
    },
    "meta": {}
}
```

Step 2: Run assertions and push result to Datahub

```
# RUNNING GE ASSERTION
run_ge_tests = BashOperator(
    task_id='pet_profiles_ge_tests_run',
    inlets=[dataset('snowflake', 'long_tail_companions.adoption.pet_profiles')],
    cwd='/usr/local/airflow/dags/long_tail_companion/02-assertion/ecommerce/',
    bash_command='/usr/local/airflow/local/bin/great_expectations checkpoint run pet_profiles',
)
```

![Datahub screenshot]
DataHub Assertions

Step 2: Verify

- Id column is distinct
- Age column is not null
- Row count > 1000 and < 2000
- Timestamp col in millis

"Are all assertions passing? Are there results in the past 12 hours?"
DataHub Assertions Circuit Breaker

```python
assertion_circuit_breaker = DatahubAssertionCircuitBreakerOperator(
    task_id="pet_profiles_assertion_circuit_breaker",
    datahub_rest_conn_id="datahub_longtail",
    urn=[
        "urn:li:dataset:
        (urn:li:dataPlatform:snowflake, long_tail_companions.adoption.pet_profiles, PROD)"
    ],
    verify_after_last_update=True,
)
```

- **Set up an Assertion Circuit Breaker Operator**
  - `assertion_circuit_breaker` is an instance of the `DatahubAssertionCircuitBreakerOperator` class.
  - `task_id` specifies the name of the task.
  - `datahub_rest_conn_id` is the connection ID for the DataHub API.
  - `urn` is a list of URNs to check the assertion status.
  - `verify_after_last_update` checks if the assertion happened after the dataset was last updated.
Another problem: Broken Data Part 2

A few weeks later...

BUG: age column set to 0 for new profiles

Prospective Adopters
Another problem: Broken Data Part 2

A few weeks later...

Bug: age column set to 0 for new profiles
Another problem: Broken Data Part 2

A few weeks later...

Tests can't catch everything
DataHub Incidents

Step 1: Raise Incident

There are 2 active incidents:
- **Data Backfill - Age set to 0 for all newly added profiles.**
  - Operational
  - Started: May 20, 2022
  - Description: Starting on May 20, 2022 new profiles were mistakenly created with age = 0 on the profiles. This is currently being backfilled.
  - Resolution: 22 May 2022 (America/Los_Angeles)
  - Status: Open
- **Incident raised because of DAO failure**
  - Operational
  - Description: Run manual_2022-05-20T12:09:32.735583+00:00 failed for dag: marketing-send_emails because task_failure
  - Resolution: 20 May 2022 by Admin
  - Status: Resolved
DataHub Incidents

Step 2: Verify

“Are there any active incidents on Pet Profiles?”
def incident_test_pre_execute(context):
    hook = DatahubRestHook("datahub_longtail")
    host, password, timeout_sec = hook._get_config()

    config = IncidentCircuitBreakerConfig(
        datahub_host=host, datahub_token=password, timeout=timeout_sec
    )
    cb = IncidentCircuitBreaker(config)
    ti = context["ti"]
    inlets = get_inlets_from_task(ti.task, context)

    for inlet in inlets:
        print(f"Checking if there is any incident for Urn: {inlet.urn}")
        if cb.is_circuit_breaker_active(inlet.urn):
            print(f"Incident Circuit Breaker is active for {inlet.urn}"),
            raise Exception(f"Incident Circuit Breaker is active for {inlet.urn}"),
        else:
            print(f"Incident Circuit Breaker is closed for {inlet.urn}"),
    return
### DAGs

<table>
<thead>
<tr>
<th>DAG</th>
<th>Owner</th>
<th>Runs</th>
<th>Schedule</th>
<th>Last Run</th>
<th>Next Run</th>
<th>Recent Tasks</th>
<th>Actions</th>
<th>Links</th>
</tr>
</thead>
<tbody>
<tr>
<td>marketing_send_emails</td>
<td>airflow</td>
<td>2</td>
<td>05:00</td>
<td>2022-05-23, 07:49:55</td>
<td>2022-05-23, 00:00:00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>snowflake_load</td>
<td>airflow</td>
<td>17</td>
<td>05:00</td>
<td>2022-05-22, 17:51:45</td>
<td>2022-05-23, 00:00:00</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Showing 1-2 of 2 DAGs
Revisiting Reliability

DataHub Incidents
DataHub Assertions
DataHub Operations

Correctness
Timeliness / Completeness
Availability

Time

At Scale

How to handle 100s of DAGs?
1000s of Datasets?
Achieving Scale: Centralizing Control

Key characteristics of a solution

- **Leverage**: Decouple Policy **Definition** from Policy **Enforcement / Evaluation**
- **Flexibility**: Seamless Policy **Evolution**
- **Configurability**: Apply targeted policies to most important assets
- **Usability**: Integration by default
DataHub Tests

Step 1: Define Tests
"All Tier1 Datasets should have passing assertions"

Step 2: Verify
"Are my inputs passing Tests?"

- Pet Profiles
- Ecommerce Airflow
- Snowflake loader
- Pet Profiles
- Email Sender
- Marketing Airflow

Tier 1
DataHub Tests

Central policy definition, distributed enforcement

Manage Tests

DataHub Tests allow you to continuously evaluate a set of conditions on the assets comprising your Metadata Graph.

- Create new test

<table>
<thead>
<tr>
<th>Name</th>
<th>Category</th>
<th>Description</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Tier 1 Datasets must have passing Assertions</td>
<td>Governance</td>
<td>All Tier 1 Datasets MUST have Assertions defined and passing.</td>
<td>1 passing, 0 failing</td>
</tr>
<tr>
<td>All Datasets must have &gt; 0 Glossary Terms</td>
<td>Governance</td>
<td>For this test, all Datasets must have a Glossary Term assigned to them.</td>
<td>1 passing, 0 failing</td>
</tr>
<tr>
<td>All Datasets must have Domain set</td>
<td>Governance</td>
<td>All datasets must have a domain set.</td>
<td>0 passing, 1 failing</td>
</tr>
<tr>
<td>All Datasets on Snowflake must have &gt; 2 Owners</td>
<td>Governance</td>
<td>Each Dataset on Snowflake MUST have &gt; 2 Owners assigned to it.</td>
<td>0 passing, 0 failing</td>
</tr>
</tbody>
</table>

Define your Test

For more information about how to configure a Test, check out the DataHub Tests Guide.

```python
<table>
<thead>
<tr>
<th>dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>fields: tags</td>
</tr>
<tr>
<td>condition: EQUALS</td>
</tr>
<tr>
<td>value: 'urn:limit:tier1'</td>
</tr>
<tr>
<td>rules:</td>
</tr>
<tr>
<td>- field: assertions</td>
</tr>
<tr>
<td>condition: EXISTS</td>
</tr>
<tr>
<td>- field: assertions.runEvents.result.type</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
```

Some tests are failing

2 passing tests, 1 failing tests

Test Results

- **Failing**
  - All Datasets must have Domain set
  - All Datasets must have > 0 Glossary Terms
- **Passing**
  - All Tier 1 Datasets must have passing Assertions
DataHub Tests Circuit Breaker

Step 1: Define Task policy in `airflow_local_settings.py`

```python
def metadata_test_pre_execute(context) -> None:
    hook = DatahubRestHook("datahub_longtail")
    host, password, timeout_sec = hook.get_config()

    config = MetadataTestCircuitBreakerConfig(
        datahub_host=host,
        datahub_token=password,
        timeout=timeout_sec,
    )
    cb = MetadataTestCircuitBreaker(config)
    print(f"context: {context}"")
    ti = context["ti"]
    inlets = get_inlets_from_task(ti.task, context)
    for inlet in inlets:
        print(f"Ur: {inlet.urn}"")
        if cb.is_circuit_breaker_active(inlet.urn):
            print(f"Circuit Breaker is active for {inlet.urn}\n"")
            raise Exception(f"Metadata Test Circuit Breaker is active for {inlet.urn}\n"")
        else:
            print(f"Metadata Test Circuit Breaker is closed for {inlet.urn}\n")
    return

def task_policy(task: BaseOperator):
    print("Applying task policy")
    task.pre_execute = metadata_test_pre_execute
```

Set up Datahub Connection
Create a Metadata Test Circuit Breaker
Check if all the metadata tests pass for all the inlets of the task
Define a task policy which get applied to every task in every dag
Realizing Reliability

Preventative Metadata: The DataHub Reliability Toolkit

DataHub Incidents
DataHub Assertions
DataHub Operations

Correctness
Timeliness / Completeness
Availability

DataHub Tests
At Scale

Acryl Data
Summary

🌟 **Data Quality** → Availability, Timeliness, Correctness

📅 **Data Reliability** → Data Quality through time

A new approach: building for Data Reliability using Metadata-driven Orchestration

How the **DataHub Operational Toolkit** can help Airflow users:

⛔️ **Operations** → availability, timeliness

✅ **Assertions + Incidents** → correctness

📝 **Tests** → achieving scale
Try Acryl DataHub

https://www.acryldata.io/sign-up
Join the MetaOps Movement

acryldata.io  
datahubproject.io  
slack.datahubproject.io  
@datahubproject
Try Open Source DataHub

```bash
> pip install acryl-datahub
```

```bash
> datahub docker quickstart
```
Acryl Data is Hiring!

Join Our Team

Join us in bringing clarity to data by enabling delightful search and discovery, data observability, and federated governance across data ecosystems.

Culture

At Acryl Data, collaboration is key, curiosity inspires action, and ambition and empathy is our (not so) secret sauce.

Values

We are a community-first, impact-driven team committed to representing the lived experiences, unique perspectives, and communities around us.
Questions?

john@acryl.io
tamas@acryl.io