

# 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

## Team

14 FTE, 3 interns, 5+ puppies



---

# Agenda

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

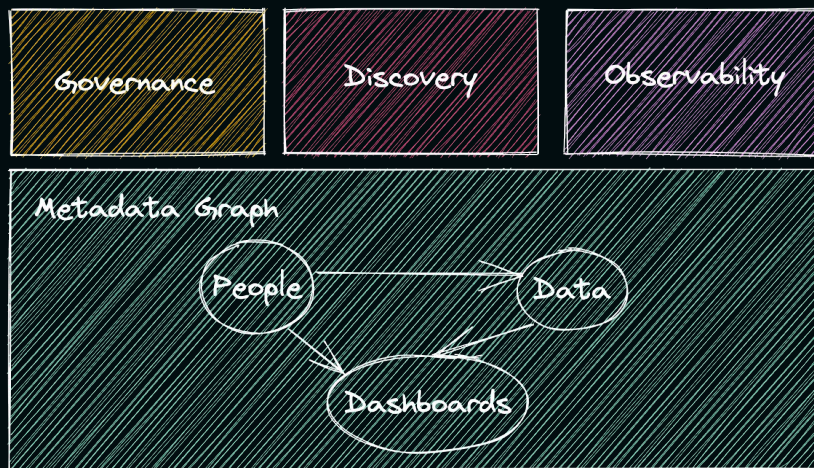




# What is DataHub?

# 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.



# What is DataHub?



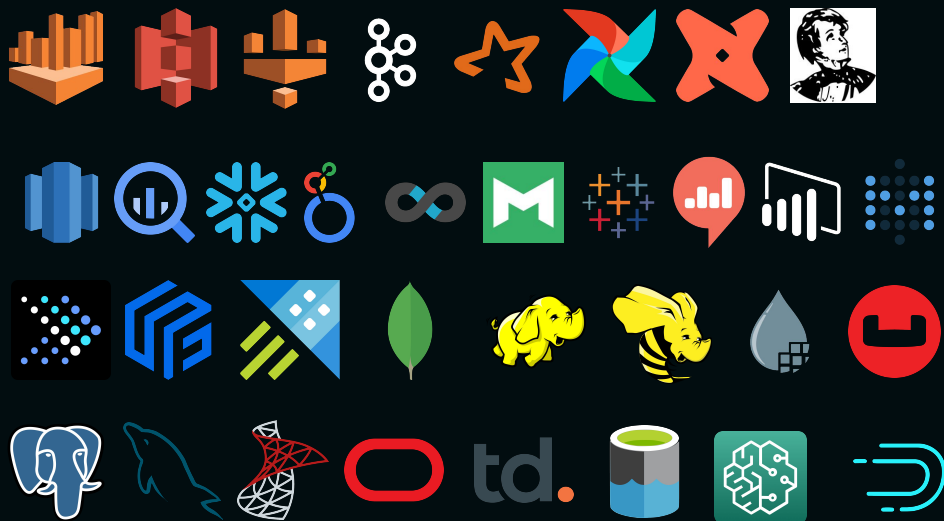
Acryl Data

See it in action! → [demo.datahubproject.io](https://demo.datahubproject.io)

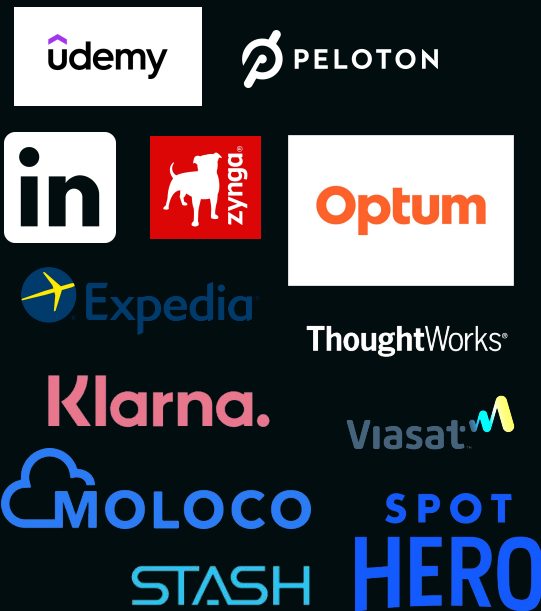
# What is DataHub?

The #1 Open Source Metadata Platform

## Integrations



## Adopters



# Community



**3,193 Slack Members**

10x YoY Growth  
Across 56 Countries & 27 Local Time Zones



## Top Member Roles



## Top Member Industries



**5.5k**

GitHub Stars



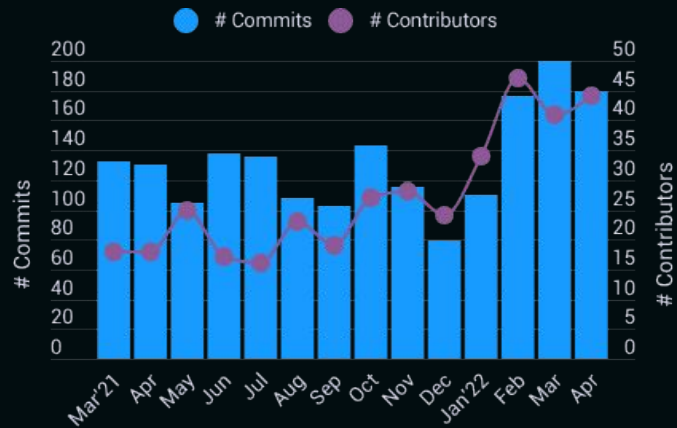
**801**

YouTube Subscribers



**397**

Blog Subscribers



# The DataHub Way

## MetaOps 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

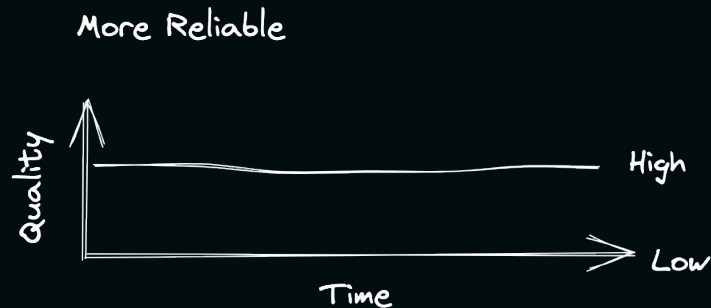
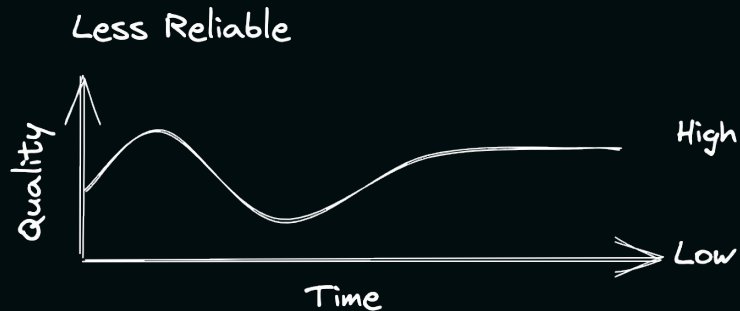


# Quality vs. Reliability

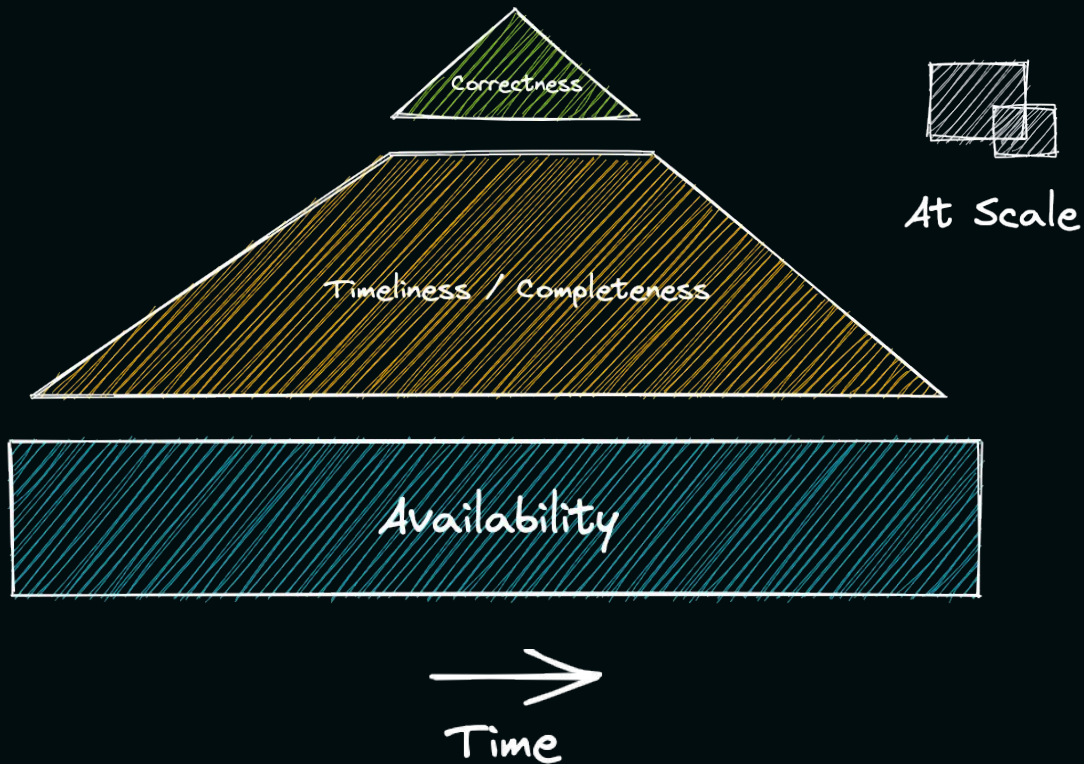
## Data Quality



## Data Reliability



# Realizing Data Reliability



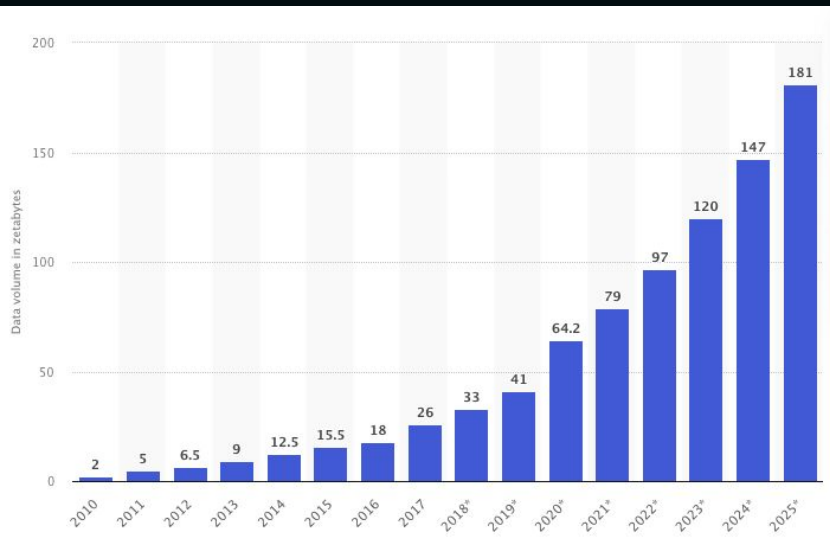
# Why should I care?

Data is becoming a **product**.

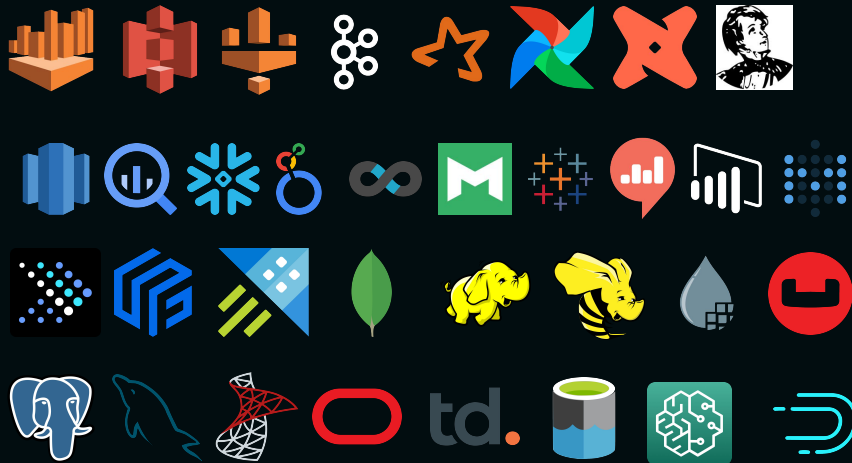


\_\_\_\_\_

# Scale



# Complexity



# Challenges

An emergent challenge: Separating **signal** from noise





# Building for Data Reliability



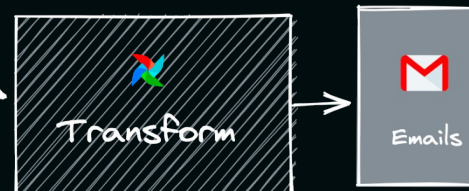
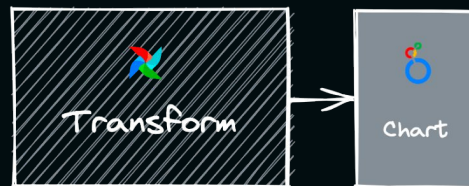
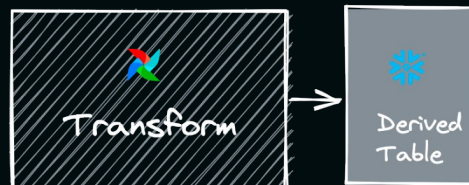
Acryl Data



Producer



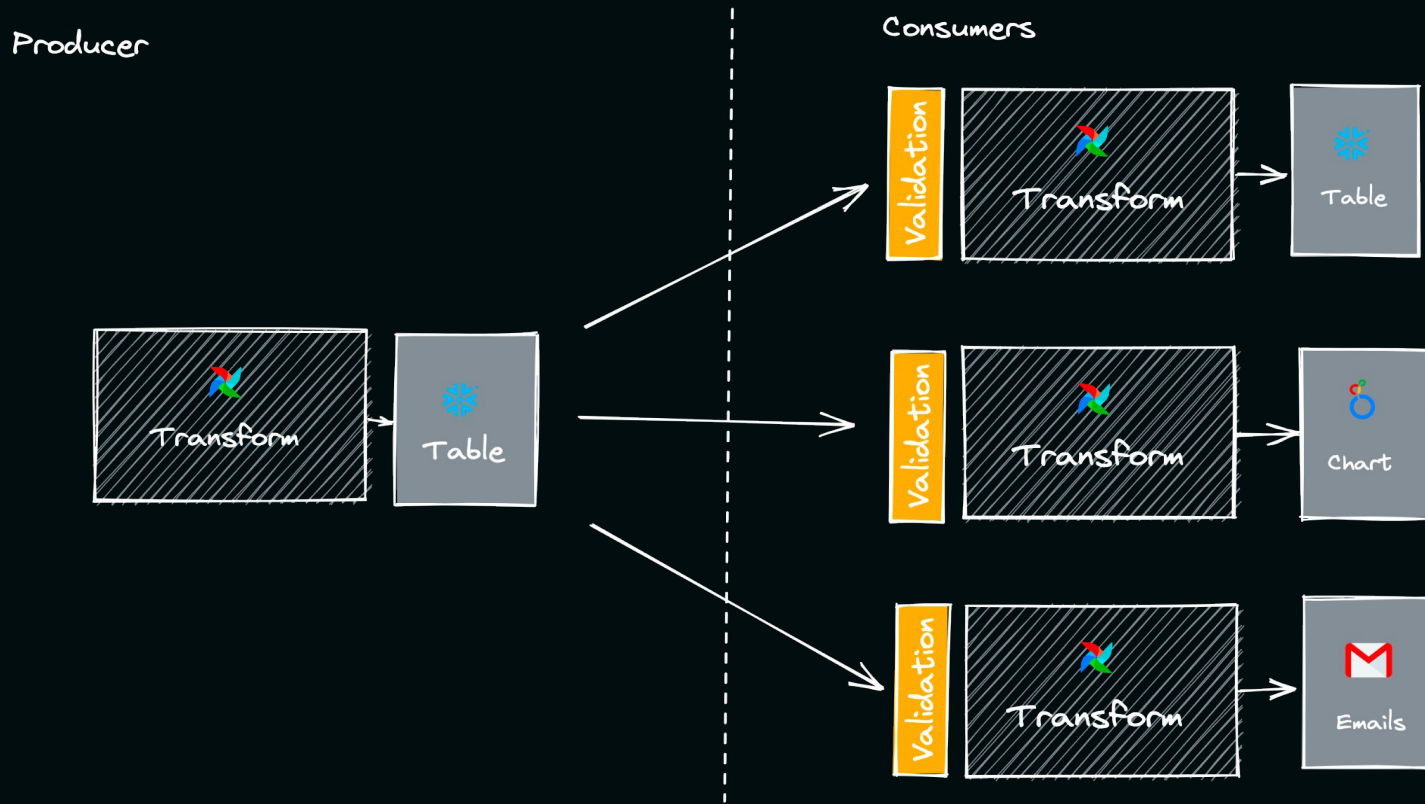
Consumers



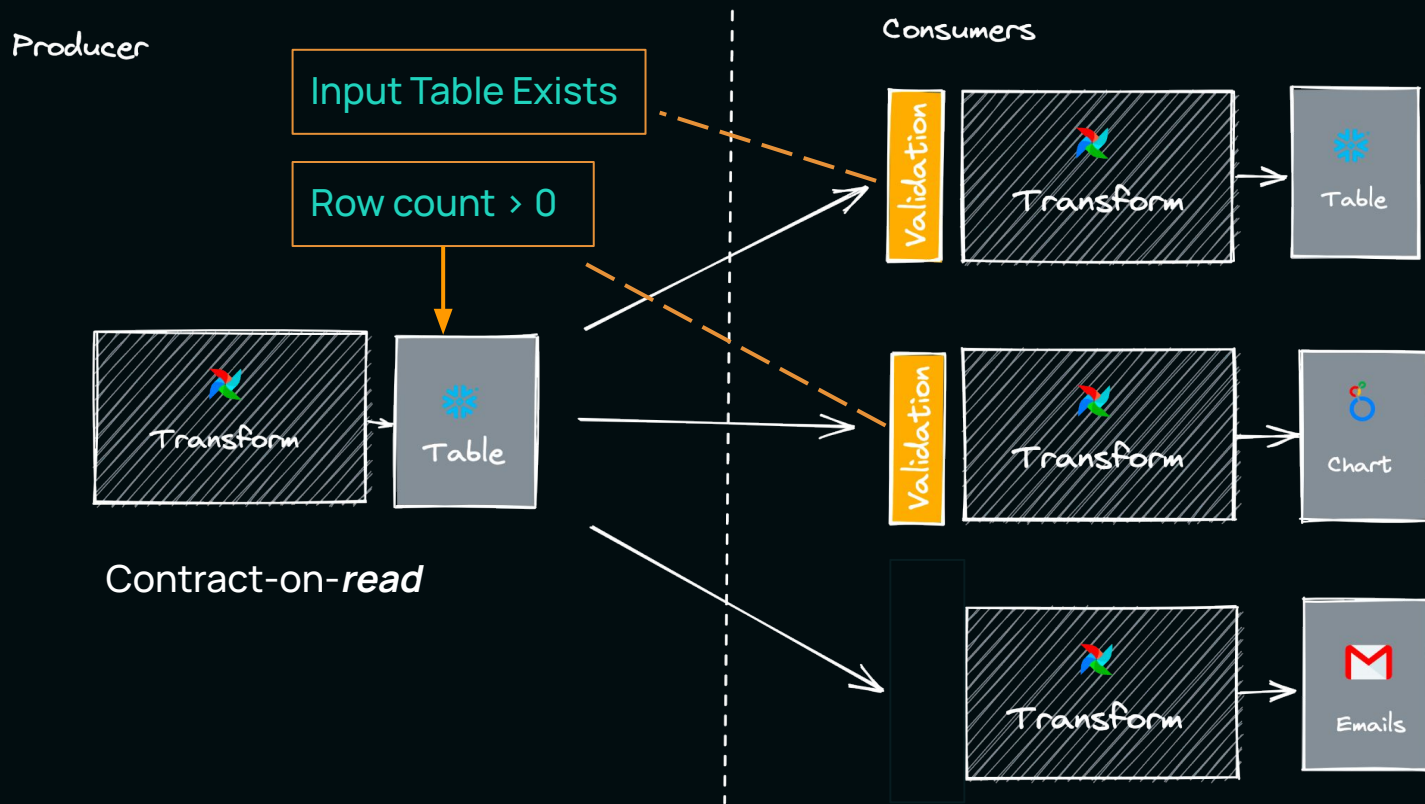
*Independent producer and consumers*



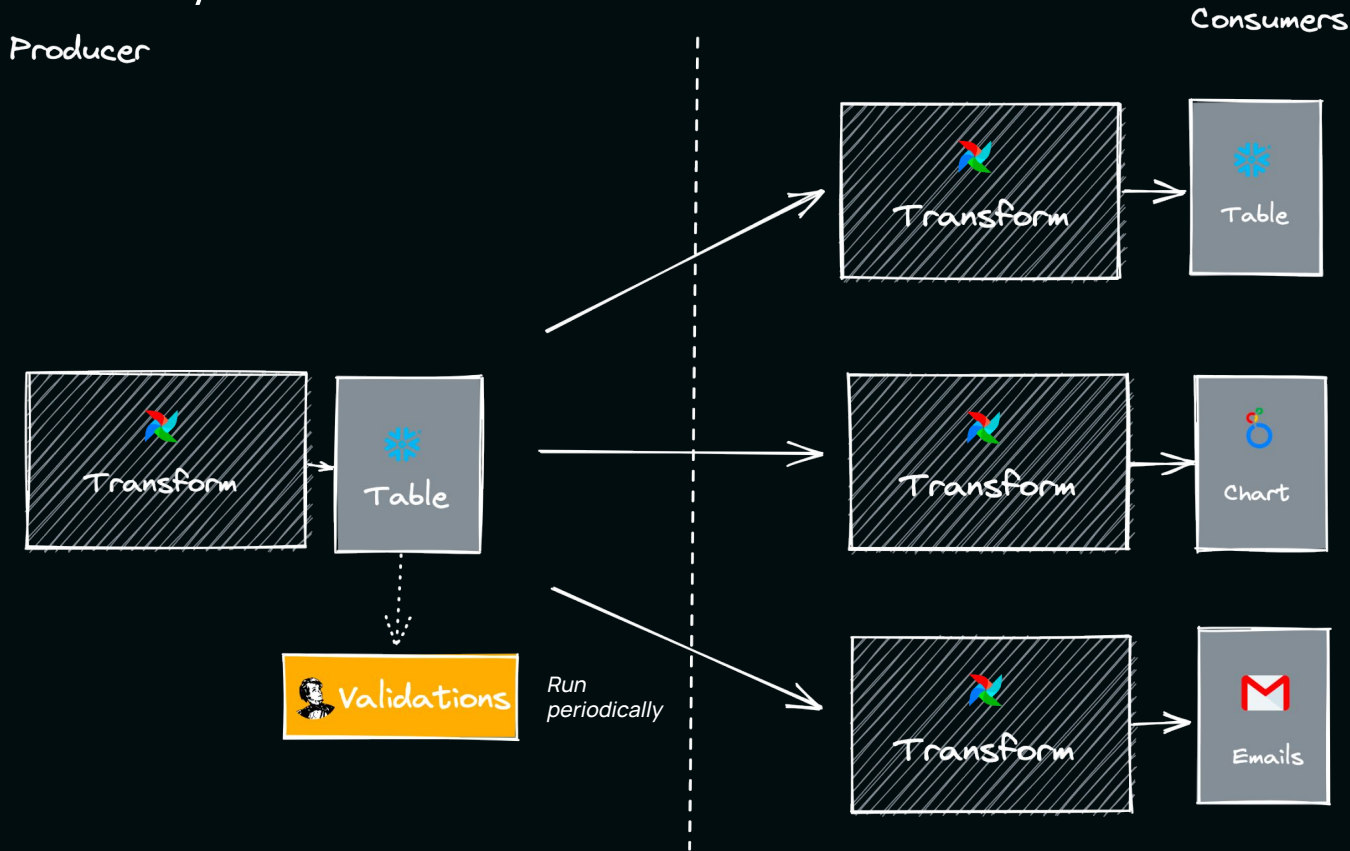
# Pattern 1: Consumer-side Validation



# Pattern 1: Consumer-side Validation



# Pattern 2: Async Validation



# Pattern 2: Async Validation

Producer

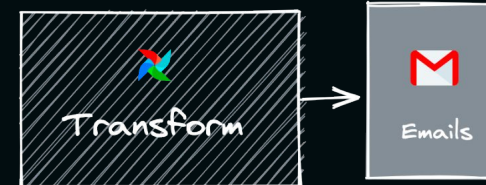
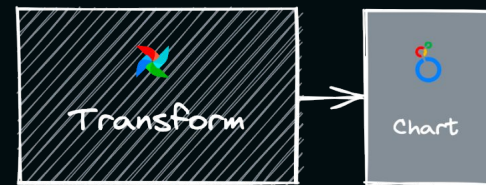
Consumers



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



Run periodically



Acryl Data

# Pattern 2: Async Validation

Producer

Consumers

Contract-*after-write*

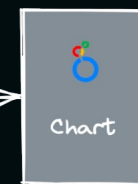
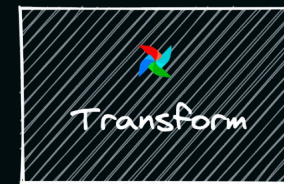


- ✓ Id column is distinct
- ✓ Age column is not null
- ✗ Row count > 1000 and < 2000
- ✓ Stdev of height column < 6.5

Validations

Alerts

Run periodically

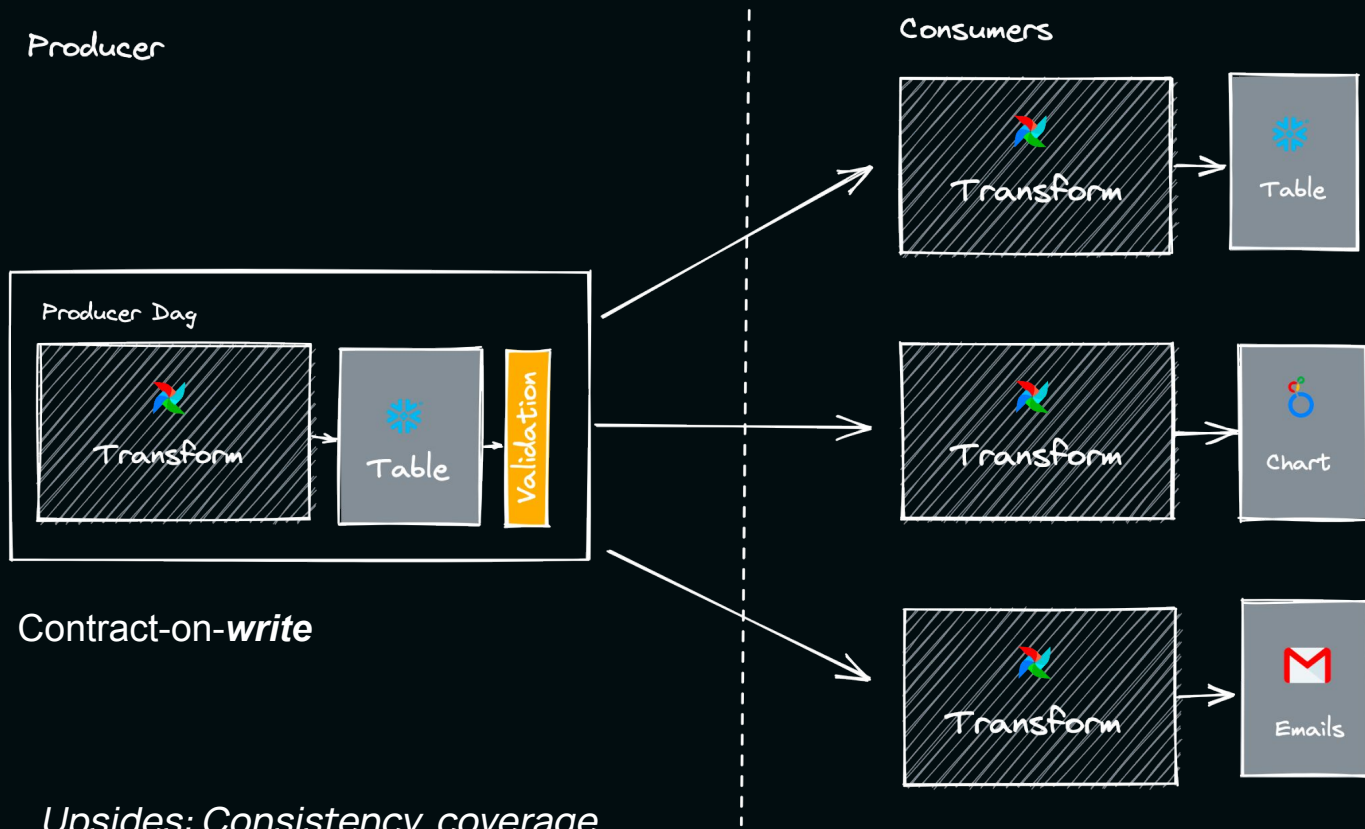


*Downside: Bad data propagates by default*



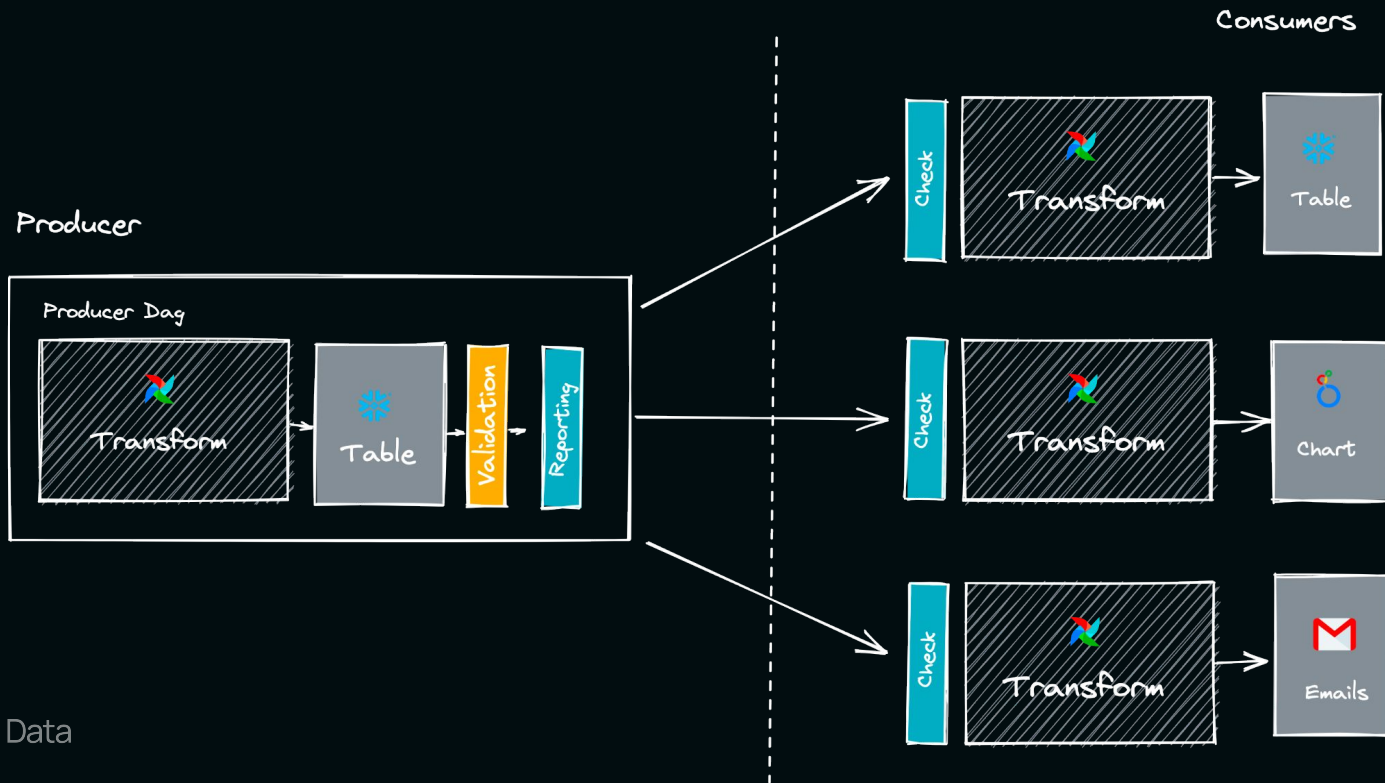
Can we do better?

# An improvement: Sync Validation



# A New Approach

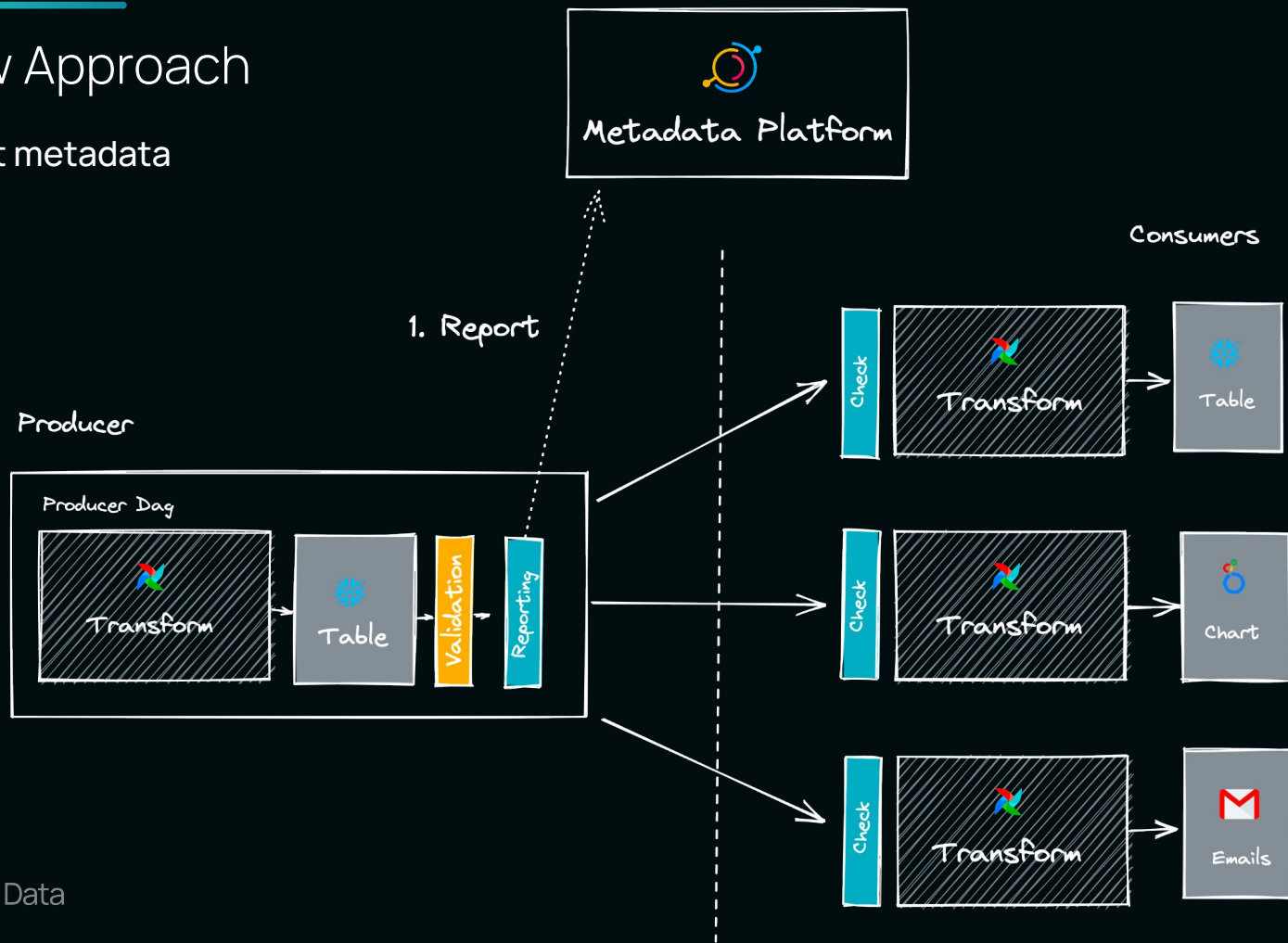
## Metadata-Driven Orchestration





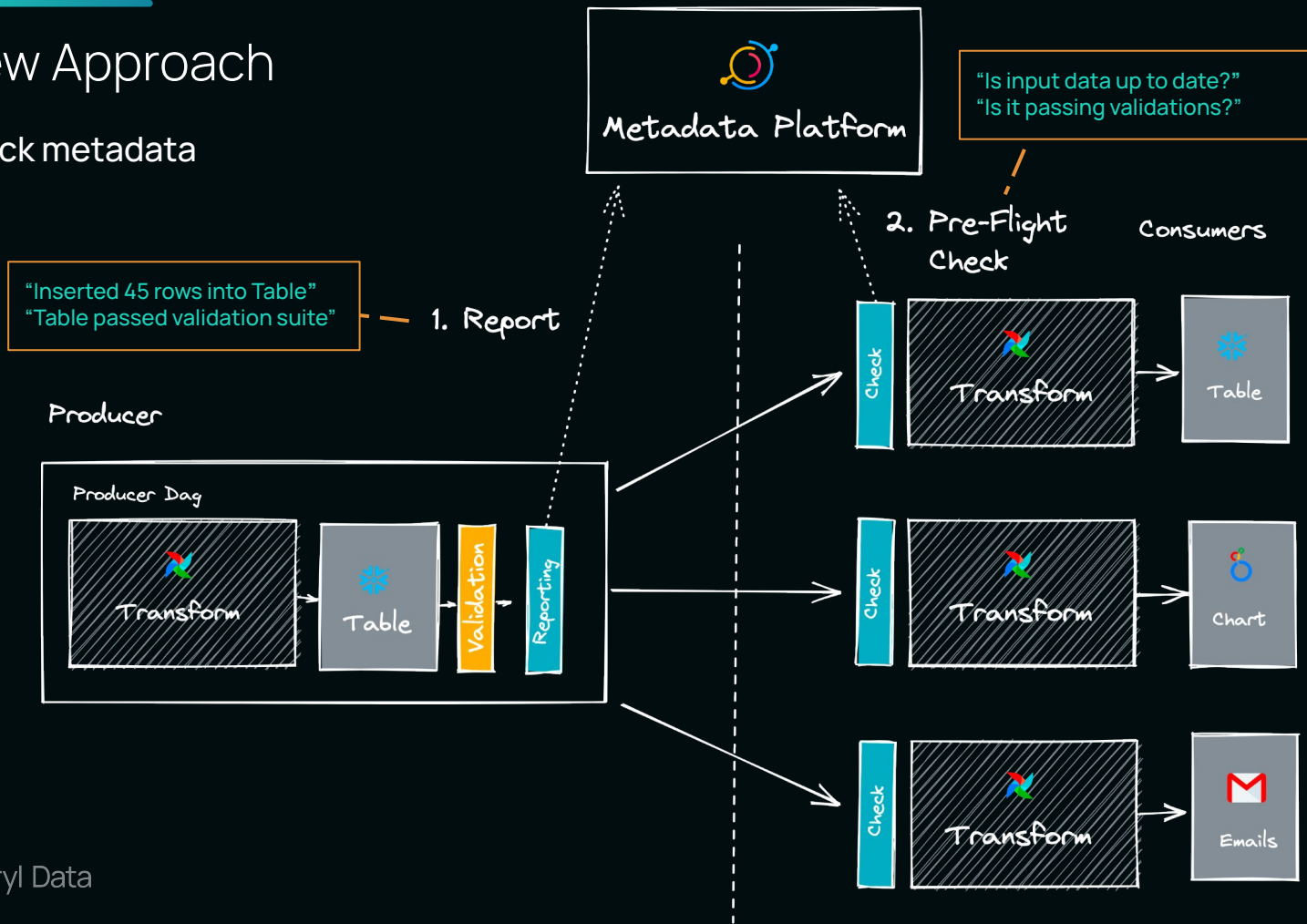
# A New Approach

## 1. Report metadata



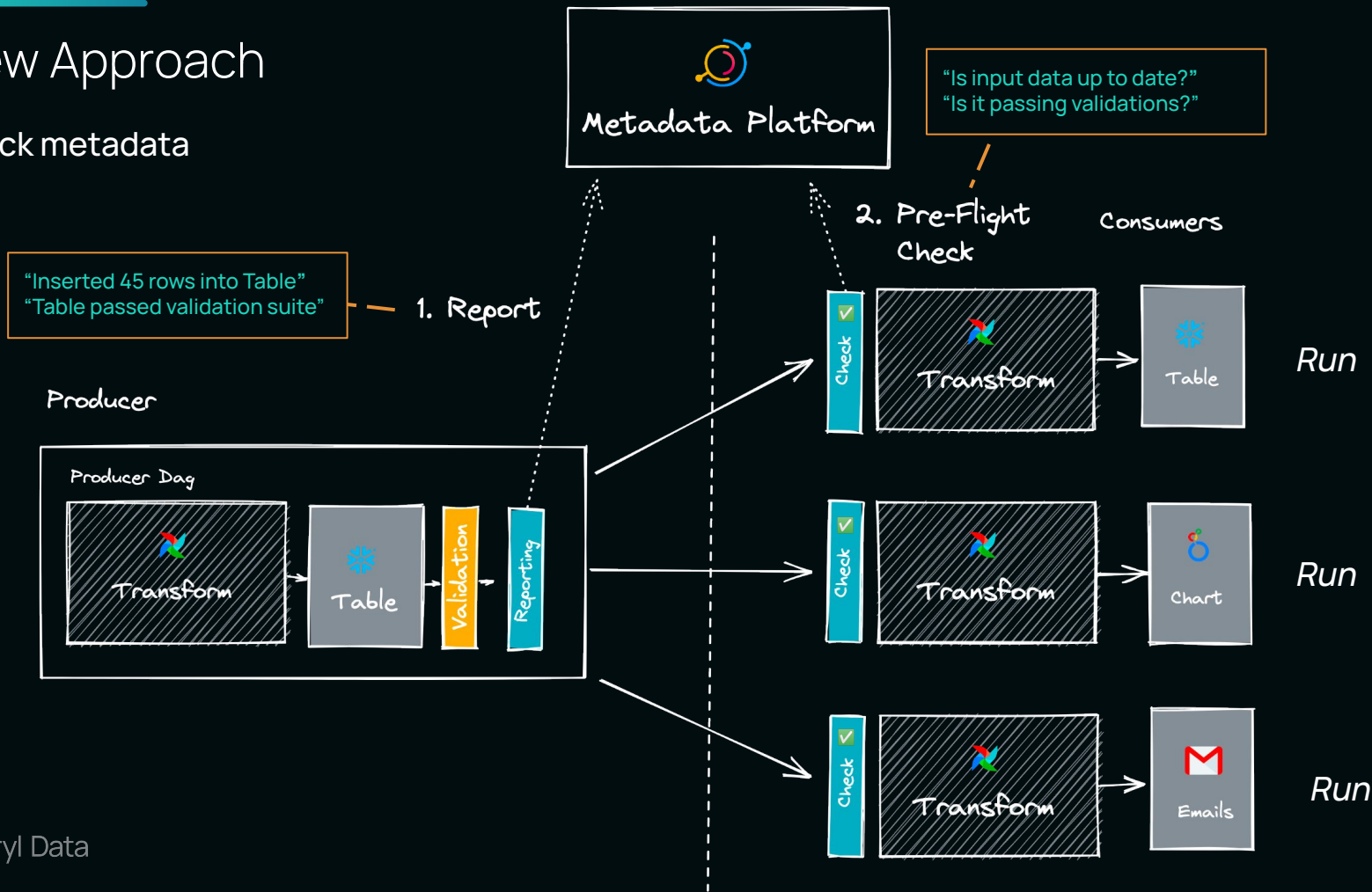
# A New Approach

## 2. Check metadata



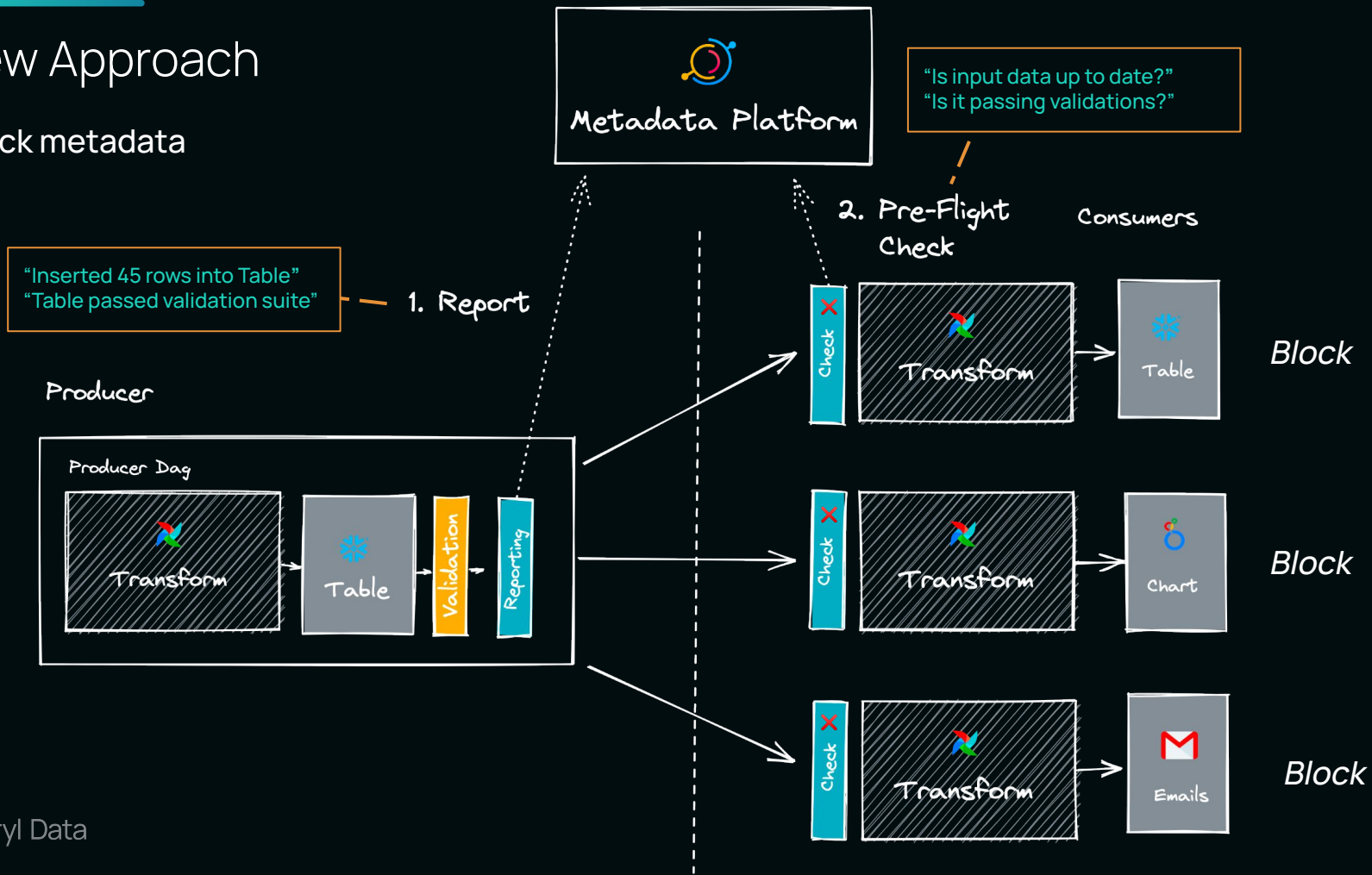
# A New Approach

## 2. Check metadata



\_\_\_\_\_

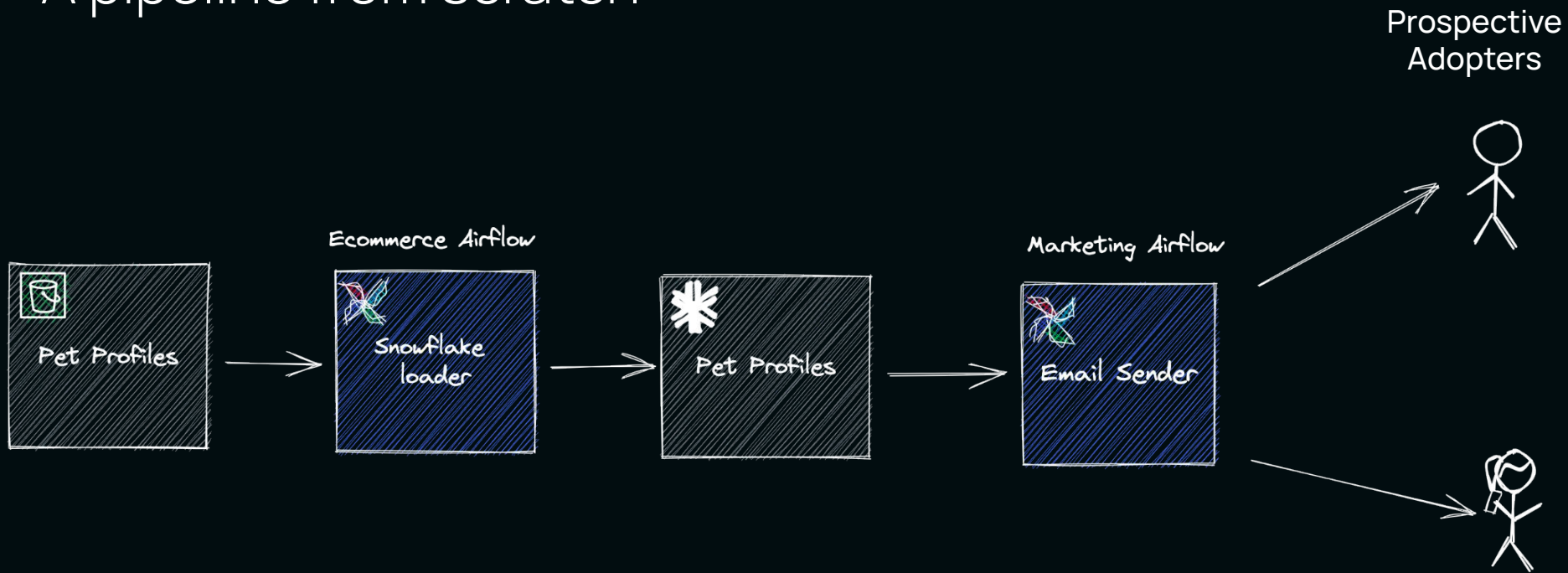
## 2. Check metadata





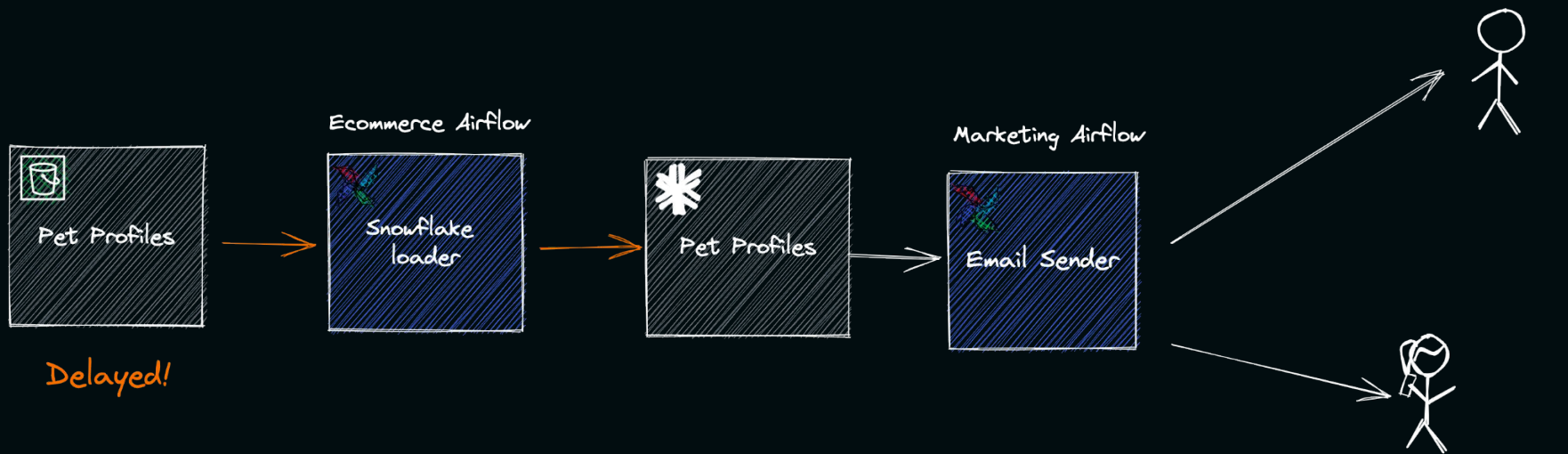
# A Practical Example

# A pipeline from scratch



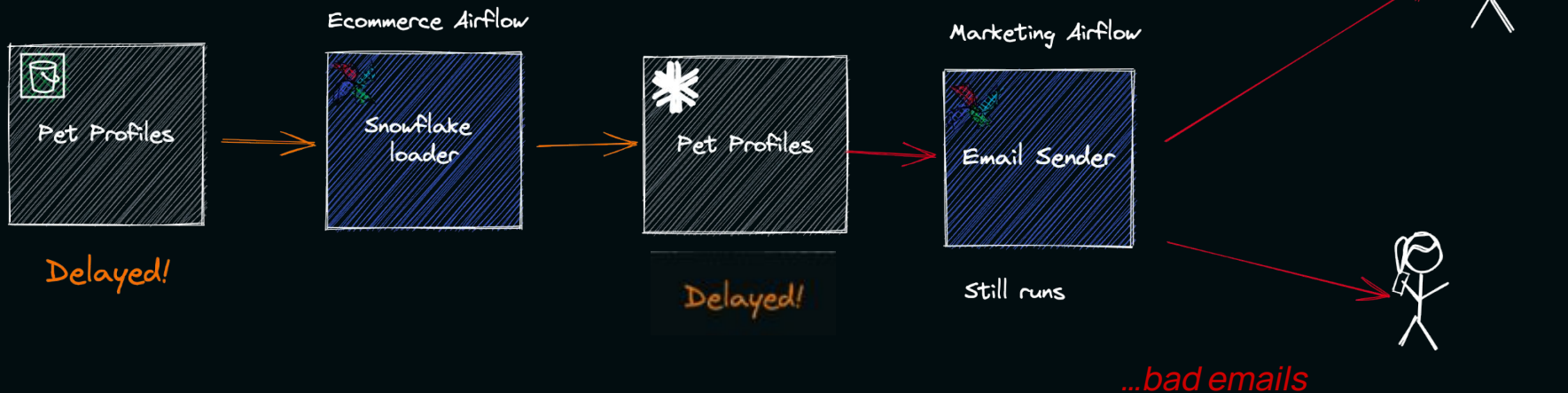
# A problem: Delayed Data

One day...



# A problem: Delayed Data

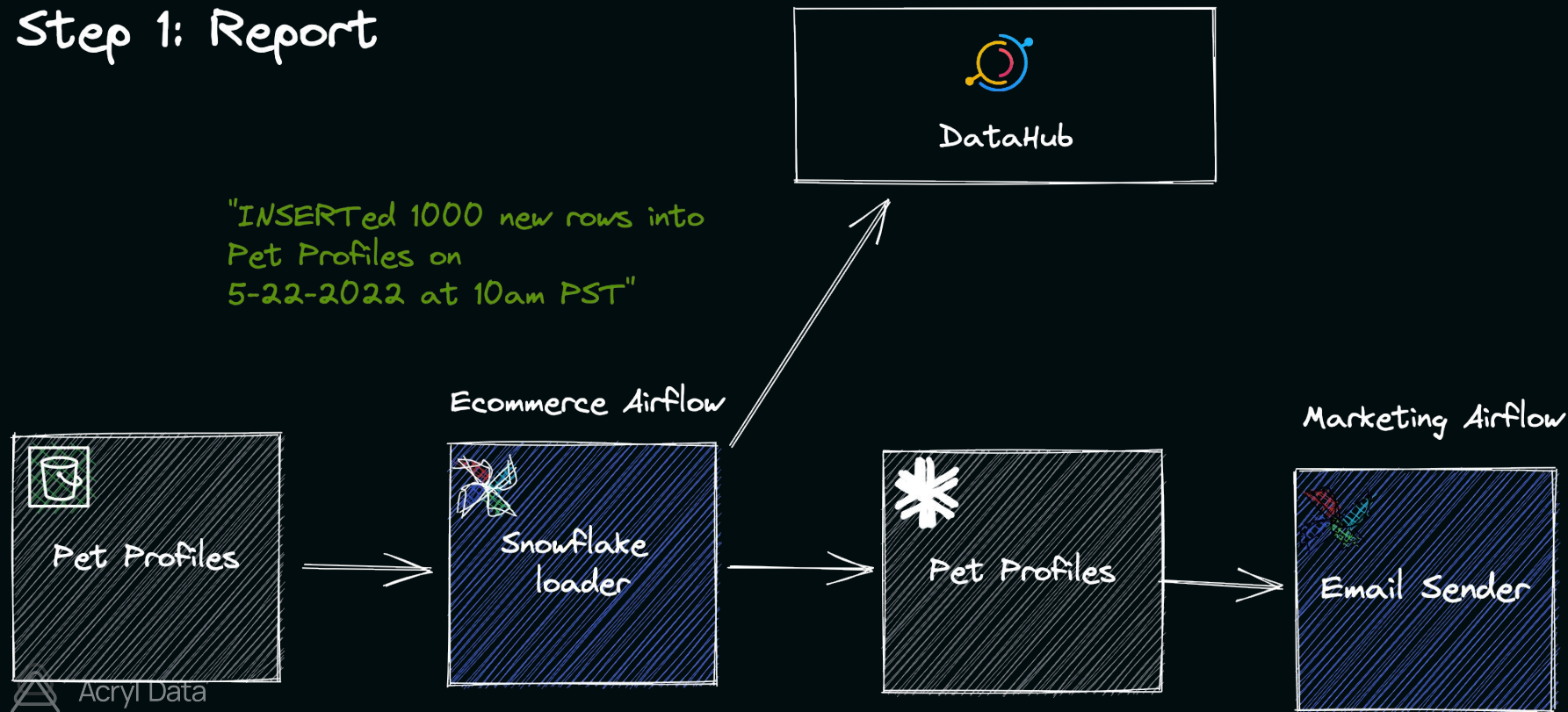
One day...





# DataHub Operations

## Step 1: Report

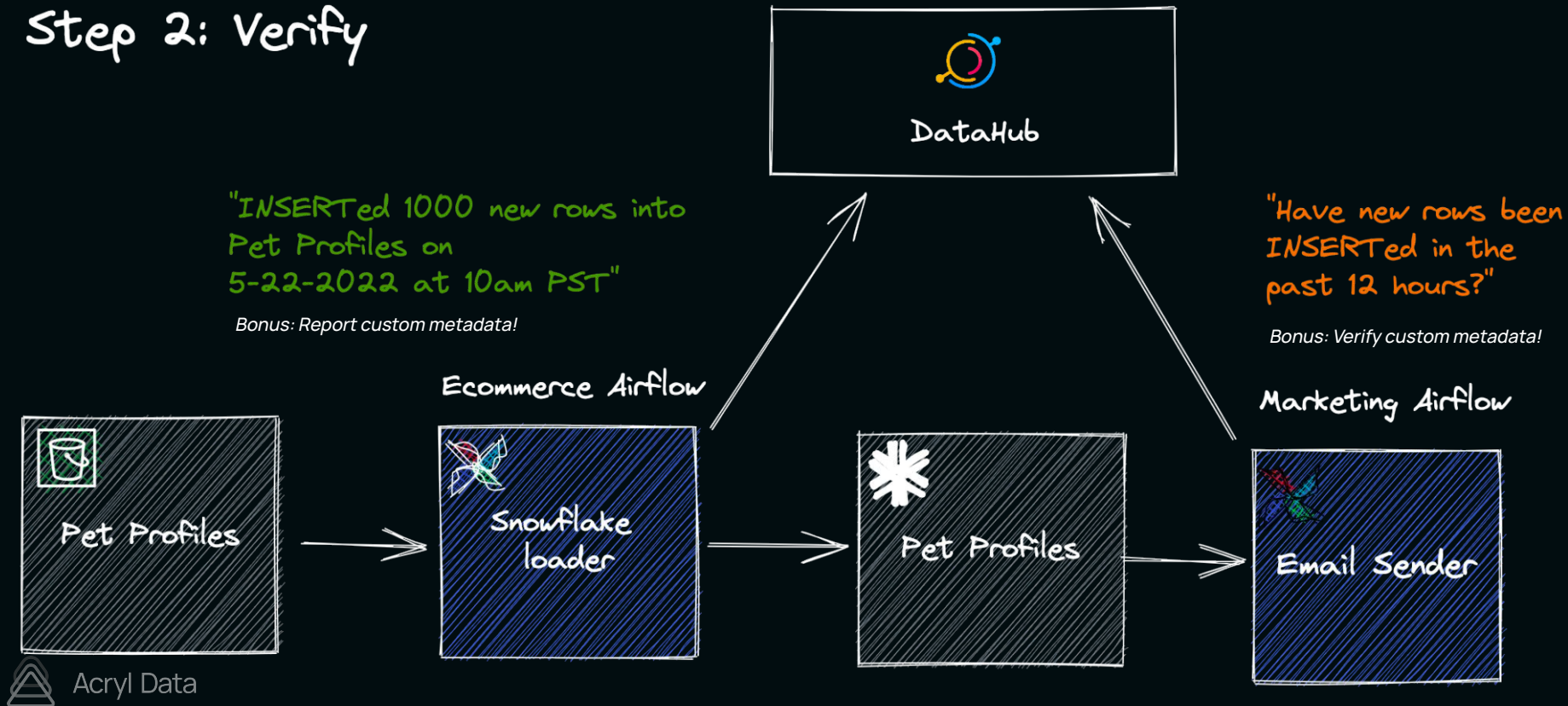


# Reporting Operations

```
def report_operation(context):  
    hook: DatahubRestHook = DatahubRestHook("datahub_longtail")  
    host, password, timeout_sec = hook._get_config()           Setup a datahub connection  
    reporter = OperationReporter(  
        datahub_host=host, datahub_token=password, timeout=timeout_sec    Create an operation reporter  
    )  
    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") Report operation data for all task outlets to Datahub  
  
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")], Define Inlets and outlets with Datahub Dataset  
    bash_command="echo Dummy Task",  
    on_success_callback=report_operation, Report operation data on success  
)
```

# DataHub Operations

## Step 2: Verify



# DataHub Operations Circuit Breaker

```
pet_profiles_operation_sensor = DatahubOperationCircuitBreakerSensor(  Set up an Operation Circuit Breaker Sensor
    task_id="pet_profiles_operation_sensor",
    datahub_rest_conn_id="datahub_longtail",
    urn=[
        "urn:li:dataset:                                     List of dataset urns to check for operation data
(urn:li:dataPlatform:snowflake,long_tail_companions.adoption.pet_profiles,PROD)"
    ],
    time_delta=datetime.timedelta(hours=12),  The time delta we expect to have operational data
)
```



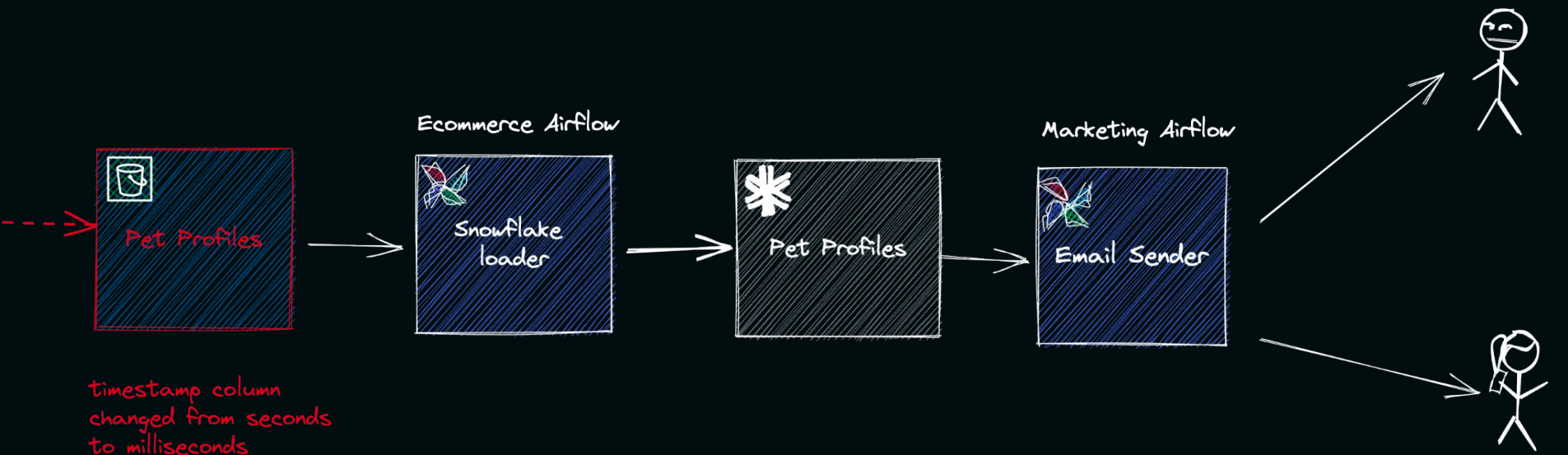
---

# Demo



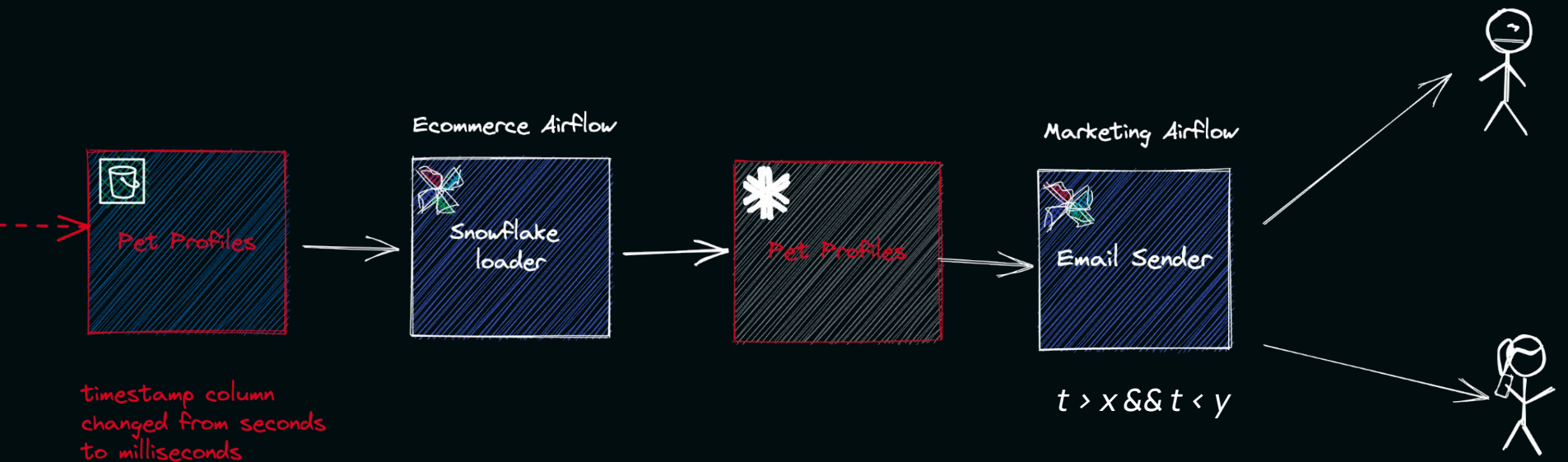
# Another problem: Broken Data

A few months later...



# Another problem: Broken Data

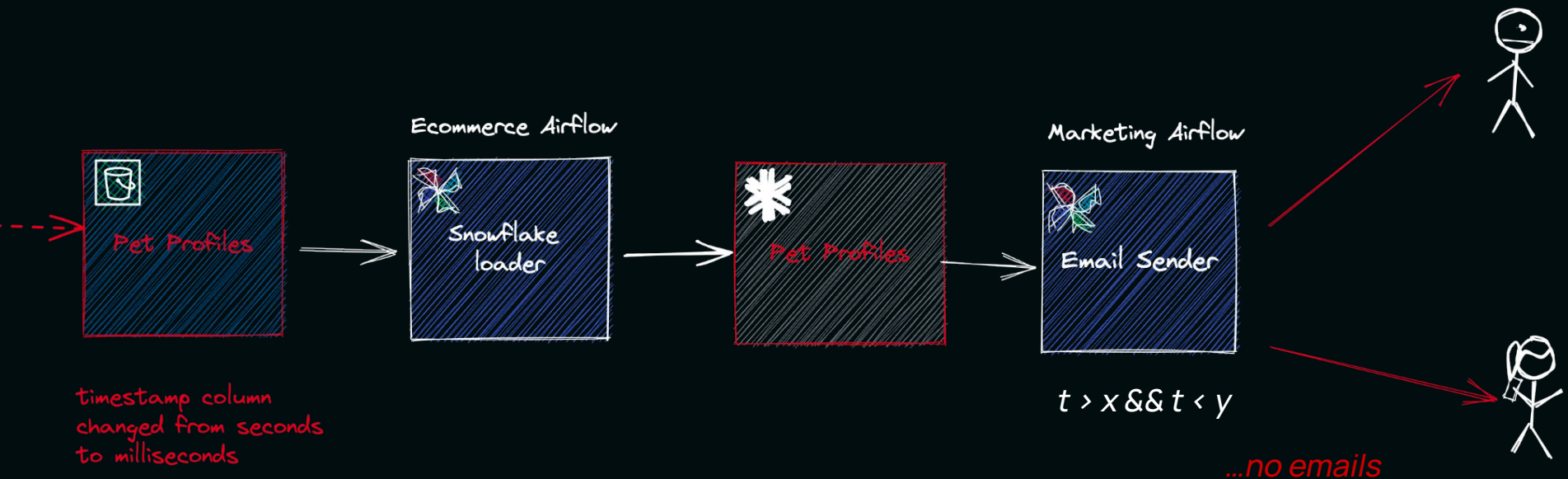
A few months later...





# Another problem: Broken Data

A few months later...





# DataHub Assertions

## Step 1: Validate + Report

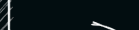
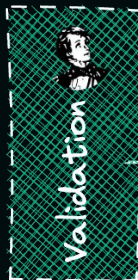
"

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

"



Ecommerce Airflow



Marketing Airflow



# Reporting Assertions

## Step 1: Define Assertions



```
{
  "expectation_type": "expect_table_row_count_to_be_between",
  "kwargs": {
    "min_value": 40000,
    "max_value": 50000
  },
  "meta": {}
},
{
  "expectation_type": "expect_column_values_to_be_in_set",
  "kwargs": {
    "column": "sex",
    "value_set": [
      "F",
      "M"
    ]
  },
  "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",
)
```

Table snowflake -> long\_tail\_companions -> adoption

pet\_profiles ✓ ⚠

Schema Documentation Properties Lineage Queries Stats Validation Incidents

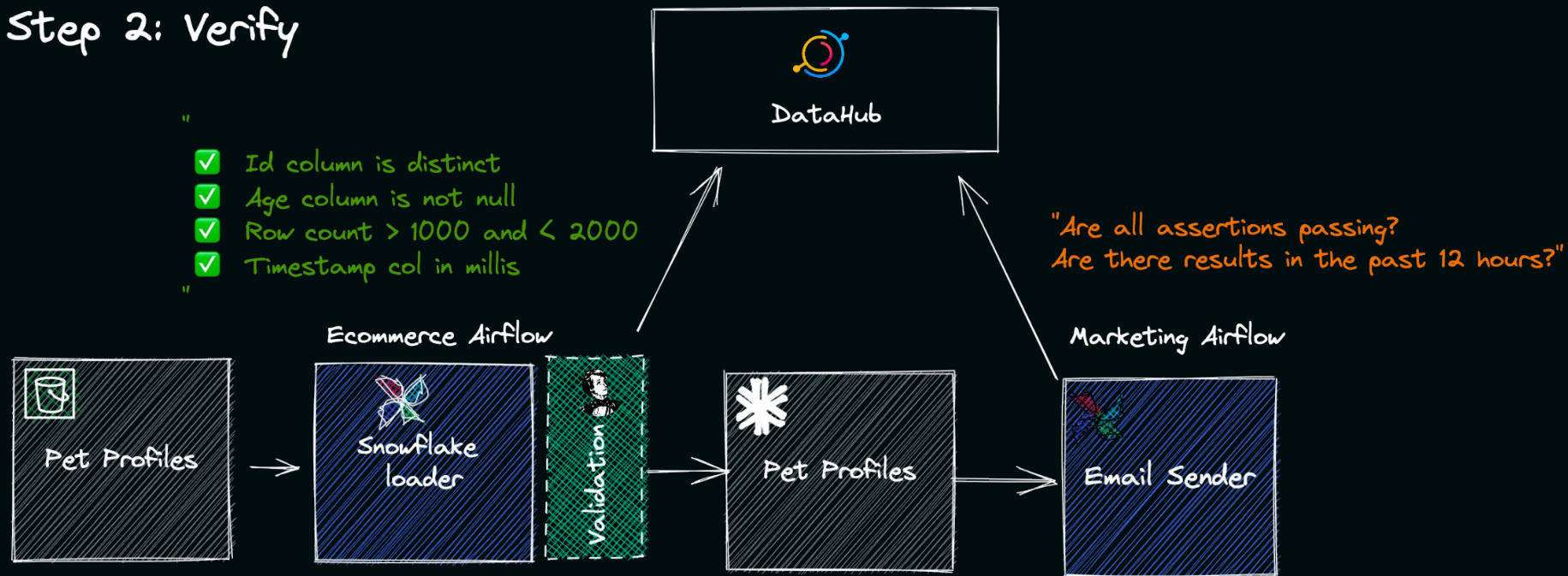
Assertions (8) Tests (3)

All assertions have passed  
8 successful assertions, 0 failed assertions

✓ Passed	Column species values are not null	🔍	❌
✓ Passed	Column age_m values are not null	🔍	❌
✓ Passed	Column age_y values are not null	🔍	❌
✓ Passed	Column spay_neutered values are not null	🔍	❌
✓ Passed	Column sex values are in ["F", "M"]	🔍	❌
✓ Passed	Column sex values are not null	🔍	❌
✓ Passed	Dataset columns are equal to ["profile_id", "species", "breed", "sex", "color", "coat_type", "name", "age_m", "age_y", "spay_neutered", "temperament", "created_at", "updated_at"]	🔍	❌
✓ Passed	Dataset row count is between 40,000 and 50,000	🔍	❌

# DataHub Assertions

## Step 2: Verify



# DataHub Assertions Circuit Breaker

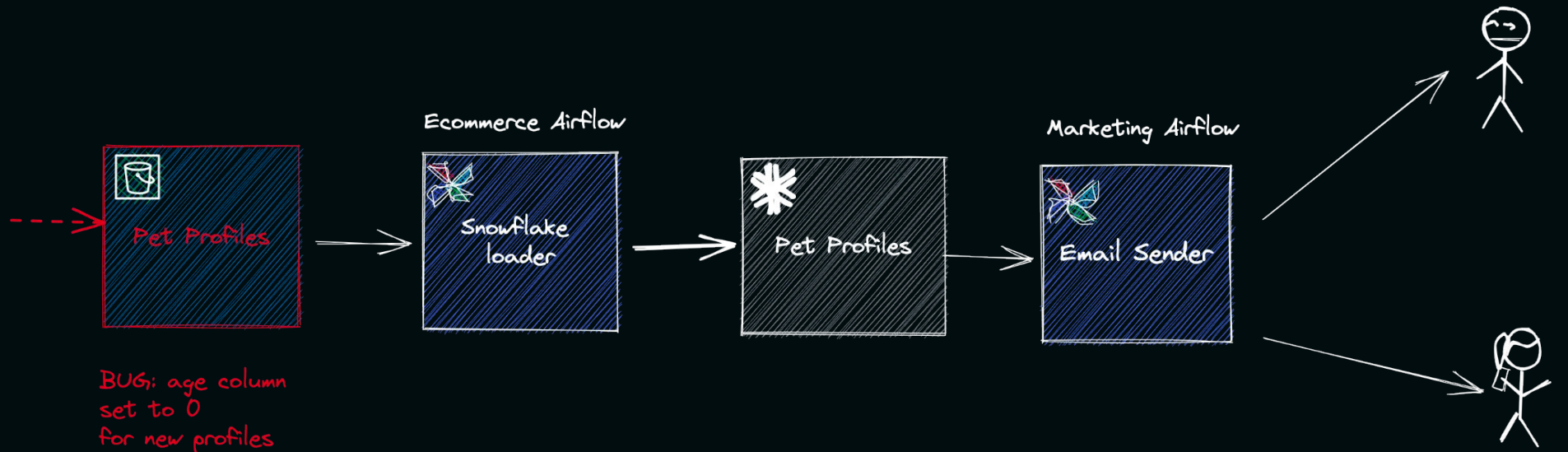
```
assertion_circuit_breaker = DatahubAssertionCircuitBreakerOperator(  Set up an Assertion Circuit Breaker Operator
    task_id="pet_profiles_assertion_circuit_breaker",
    datahub_rest_conn_id="datahub_longtail",
    urn=[
        "urn:li:dataset:                                List of urns we want to check assertion status
(urn:li:dataPlatform:snowflake,long_tail_companions.adoption.pet_profiles,PROD)"
    ],
    verify_after_last_update=True,                        If we want to check assertion happened after
                                                         the dataset was last updated
)
```

---

# Demo

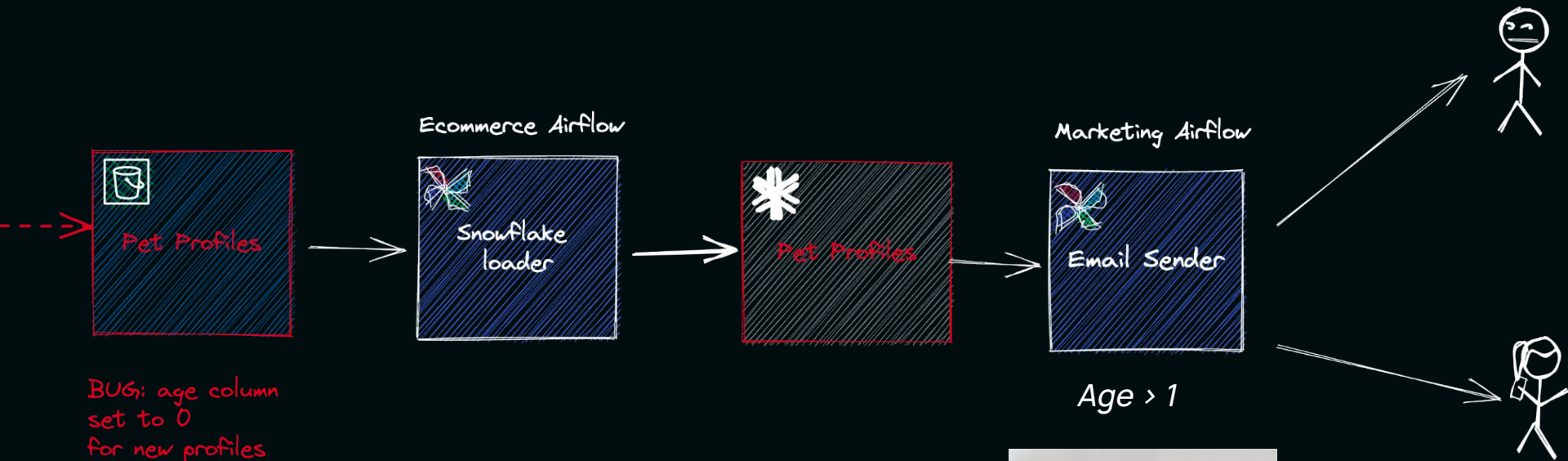
# Another problem: Broken Data Part 2

A few weeks later...



# Another problem: Broken Data Part 2

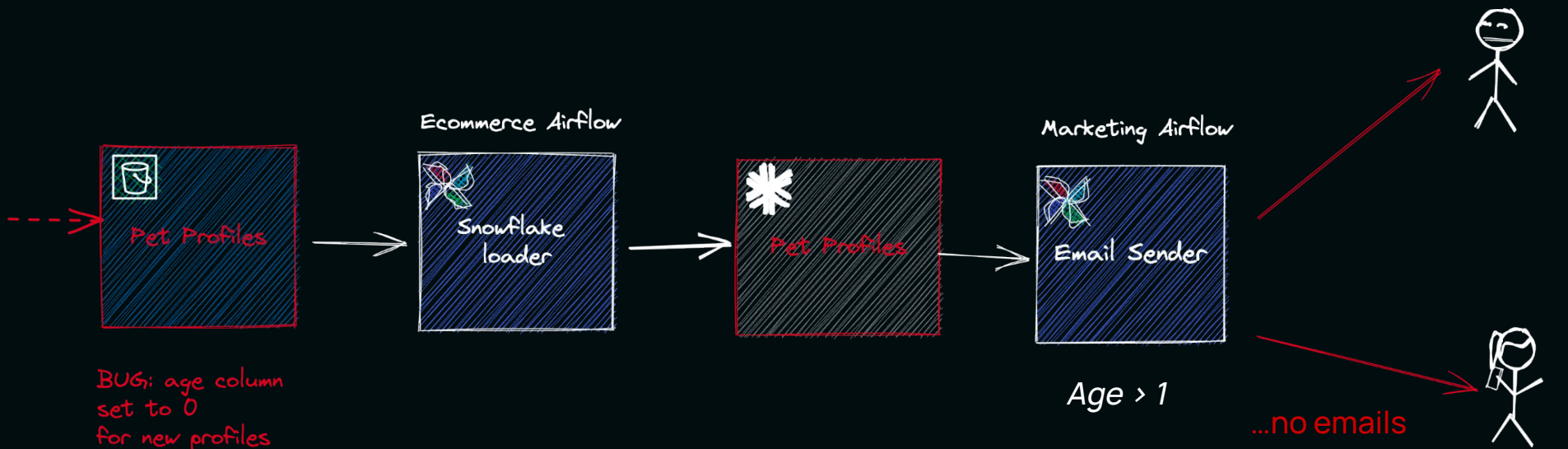
A few weeks later...





# Another problem: Broken Data Part 2

A few weeks later...



Tests can't catch everything



# DataHub Incidents

## Step 1: Raise Incident

+ Raise Incident

ions > adoption > pet\_profiles

tion

Quer

Operati

with age

Raise Incident

Type

Operational

\* Title

Data Backfill - Age set to 0 for all newly added profiles.

\* Description

Starting on May 20, 2022 new profiles were c

Cancel

Add

Table Snowflake > long\_tail\_companions > adoption

pet\_profiles ✓ 📄 ⚠️

Schema Documentation Properties Lineage Queries Stats Validation Incidents

+ Raise Incident All

⚠️ **There are 2 active incidents**  
2 active incidents, 2 resolved incidents

**Data Backfill - Age set to 0 for all newly added profiles.** Operational

Starting on May 20, 2022 new profiles were mistakenly created with age = 0 on the profiles. This is currently being backfilled.

✓ Resolve ⚠️

A 22 May 2022 (America/Los\_Angeles)

**Incident raised because of DAG failure** Operational

Run manual\_\_2022-05-20T12:09:32.735583+00:00 failed for dag: marketing-send\_emails because task\_failure

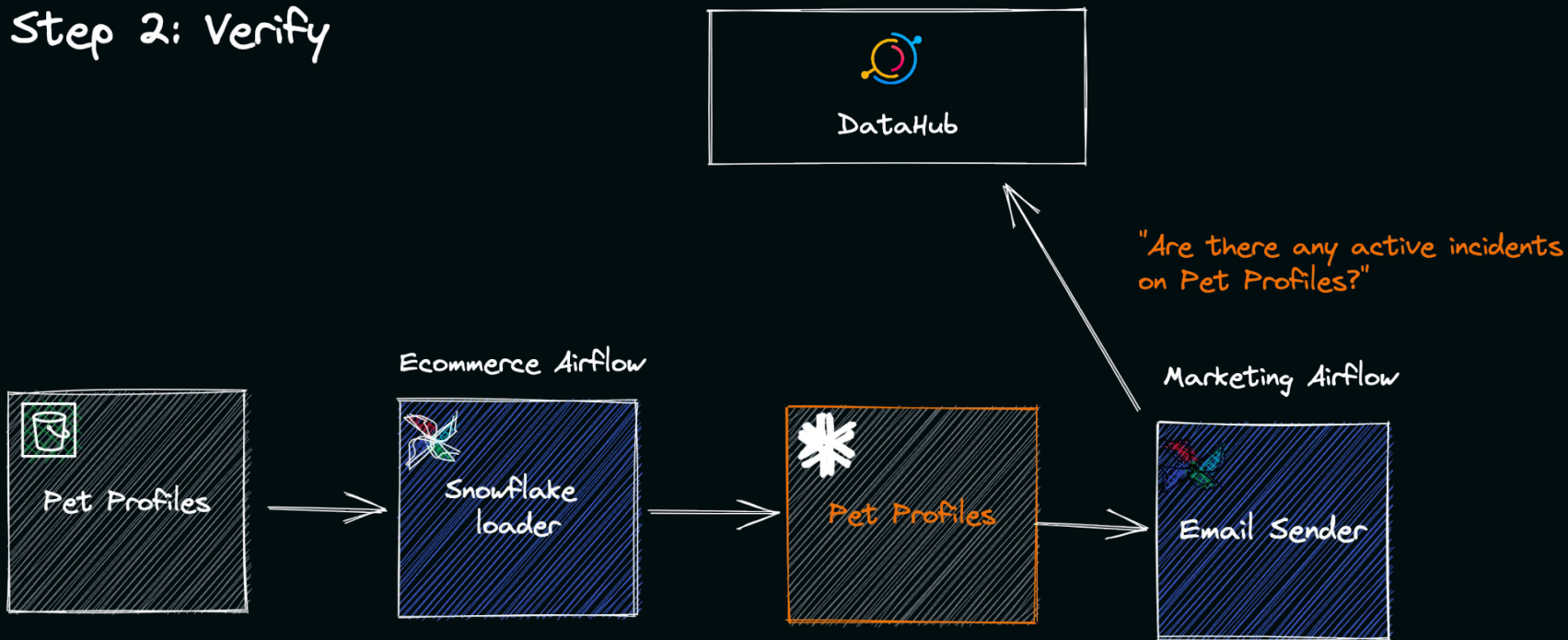
Resolved on 20 May 2022 by Admin ✓



Acryl Data

# DataHub Incidents

## Step 2: Verify




# DataHub Incidents Circuit Breaker

```
def incident_test_pre_execute(context):  
    hook: DatahubRestHook = DatahubRestHook("datahub_longtail")    Set up a Datahub Connection  
    host, password, timeout_sec = hook._get_config()  
  
    config: IncidentCircuitBreakerConfig = IncidentCircuitBreakerConfig(  
        datahub_host=host, datahub_token=password, timeout=timeout_sec  
    )  
    cb = IncidentCircuitBreaker(config)    Define an Incident Circuit Breaker  
    ti = context["ti"]  
    inlets = get_inlets_from_task(ti.task, context)    Get all the inlets for the task  
    for inlet in inlets:  
        print(f"Checking if there is any incident for Urn: {inlet.urn}")  
        if cb.is_circuit_breaker_active(inlet.urn):    Circuit break on any active incident  
            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
```



# Demo

 Airflow

Security - Browse - Admin - Docs -

07:50 UTC - [Log In](#)

DAGs

All 2 Active 2 Paused 0

Filter DAGs by tag

Search DAGs

# Revisiting Reliability

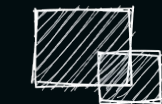
DataHub Incidents



DataHub Assertions



DataHub Operations



At Scale



How to handle 100s  
of DAGs?  
1000s of Datasets?



Acryl Data



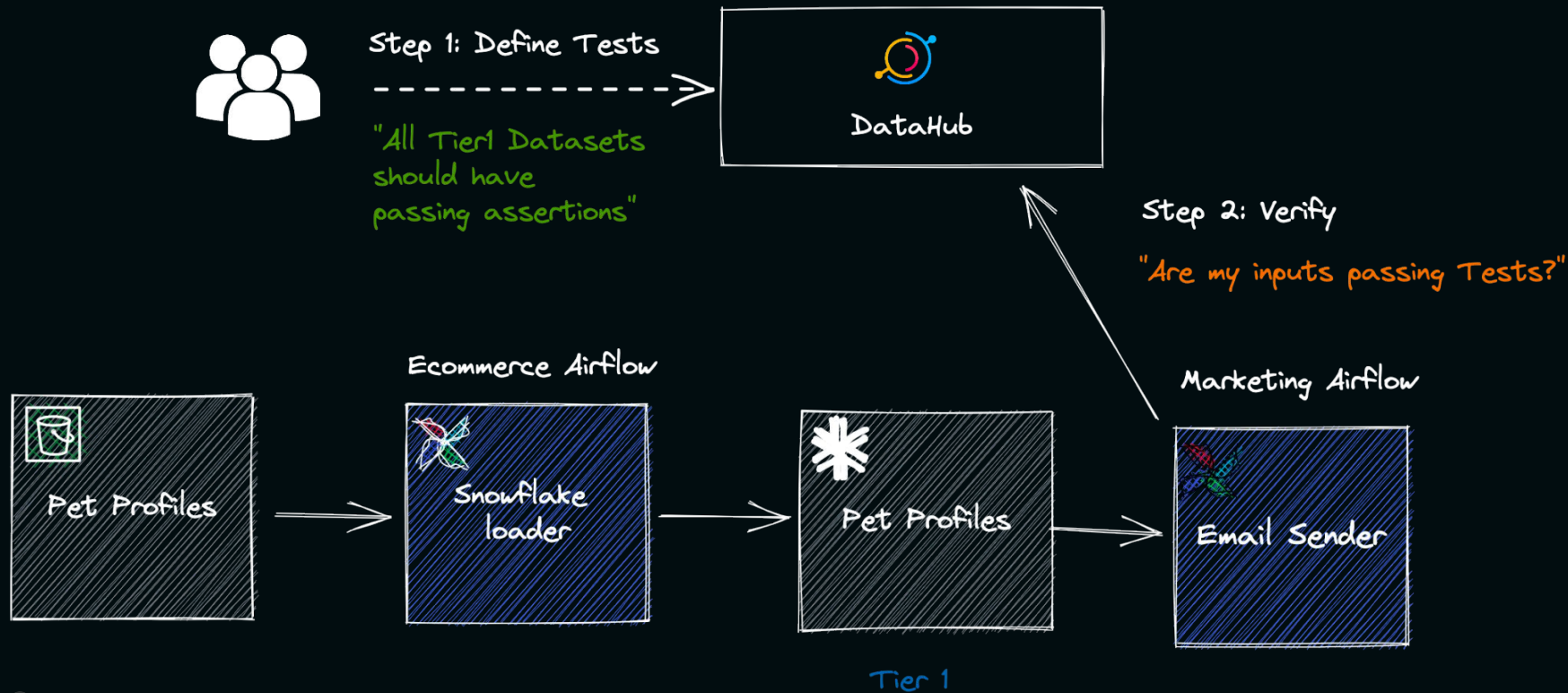
---

# 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



## Central policy definition, distributed enforcement

\* Name

Give your test a name.

All Tier 1 Datasets must have passing Assertions

\* Category

The category of your test.

Governance

Description

An optional description to help keep track of your test.

All Tier 1 Datasets MUST have Assertions defined and passing.

### Define your Test

For more information about how to configure a Test, check out the [DataHub Tests Guide](#).


```
1  on:
2    dataset:
3      -
4        field: tags
5        condition: EQUALS
6        values:
7          - 'urn:li:tag:Tier1'
8  rules:
9    -
10     field: assertions
11     conditon: EXISTS
12   -
13     field: assertions.runEvents.result.type
14     condition: NOT_EQUALS
15     value: FAILURE
16
```

## Manage Tests

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

[+ Create new test](#)

Name	Category	Description	Results
All Tier 1 Datasets must have passing Assertions	Governance	All Tier 1 Datasets MUST have Assertions defined and passing.	1 passing, 0 failing
All Datasets must have > 0 Glossary Terms	Governance	For this test, all Datasets must have a Glossary Term assigned to them.	1 passing, 0 failing
All Datasets must have Domain set	Governance	All datasets must have a domain set.	0 passing, 1 failing
All Datasets on Snowflake must have > 2 Owners	Governance	Each Dataset on Snowflake MUST have > 2 Owners assigned to it.	0 passing, 0 failing



Some tests are failing

2 passing tests, 1 failing tests

Test Results

Failing

All Datasets must have Domain set

Governance All datasets must have a domain set.

Passing

All Datasets must have > 0 Glossary Terms

Governance For this test, all Datasets must have a Glossary Term assigned to them.

Passing

All Tier 1 Datasets must have passing Assertions

Governance All Tier 1 Datasets MUST have Assertions defined and passing.



# DataHub Tests Circuit Breaker

## Step 1: Define Task policy in *airflow\_local\_settings.py*

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

Set up Datahub Connection

```
    config: MetadataTestCircuitBreakerConfig = MetadataTestCircuitBreakerConfig(
        datahub_host=host,
        datahub_token=password,
        timeout=timeout_sec,
    )
```

Create a Metadata Test Circuit Breaker

```
    cb = MetadataTestCircuitBreaker(config)
    print(f"context: {context}")
    ti = context["ti"]
    inlets = get_inlets_from_task(ti.task, context)
    for inlet in inlets:
        print(f"Urn: {inlet.urn}")
        if cb.is_circuit_breaker_active(inlet.urn):
            print(f"Circuit Breaker is active for {inlet.urn}")
            raise Exception(f"Metadata Test Circuit Breaker is active for {inlet.urn}")
        else:
            print(f"Metadata Test Circuit breaker is closed for {inlet.urn}")
    return
```

Check if all the metadata tests pass for all the inlets of the task

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

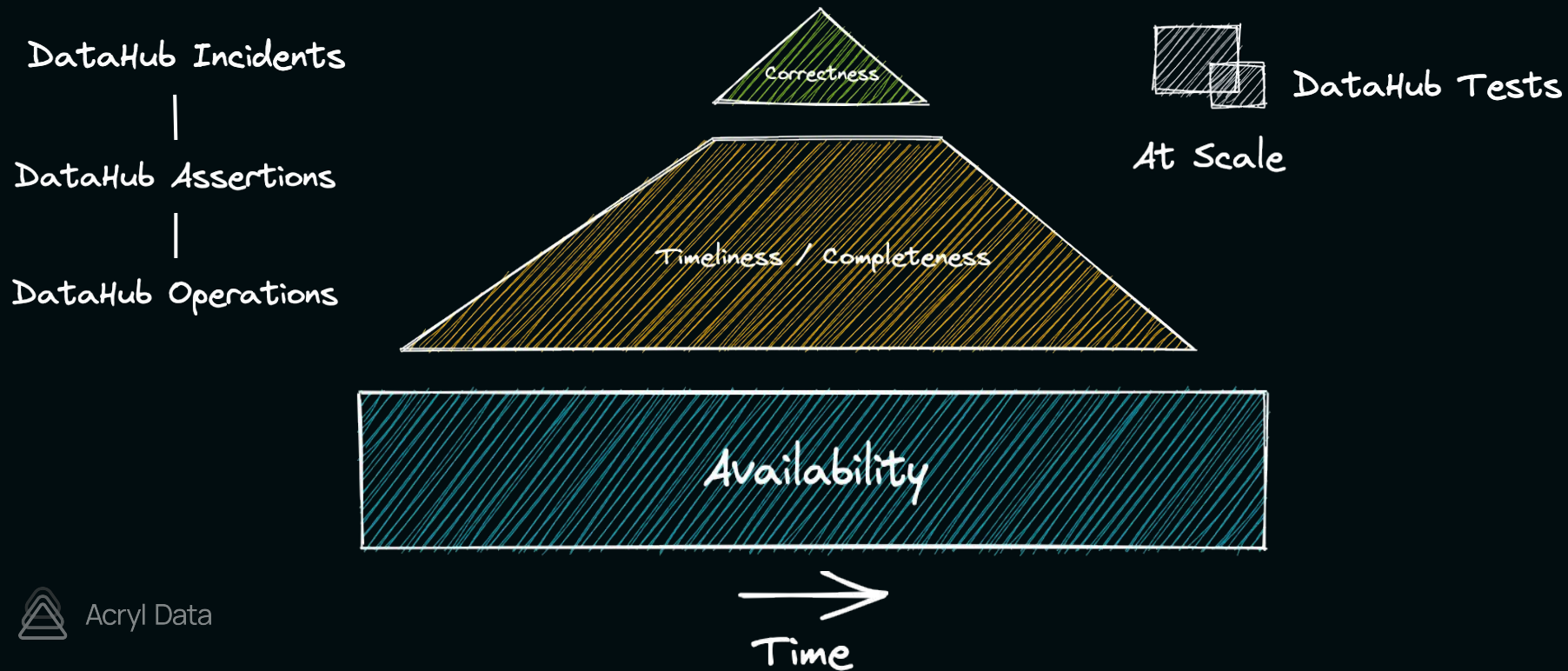
Define a task policy which get applied to every task in every dag

---

# Demo

# Realizing Reliability

Preventative Metadata: The DataHub Reliability Toolkit



# 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>



Acryl Data

# Join the MetaOps Movement

[acryldata.io](https://acryldata.io)

[datahubproject.io](https://datahubproject.io)

[slack.datahubproject.io](https://slack.datahubproject.io)

[@datahubproject](https://twitter.com/datahubproject)



Acryl Data

# Try Open Source DataHub

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

```
> datahub docker quickstart
```



# Acryl Data is Hiring!

CAREERS

## 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.



Acryl Data



# Questions?

[john@acryl.io](mailto:john@acryl.io)

[tamas@acryl.io](mailto:tamas@acryl.io)



Acryl Data