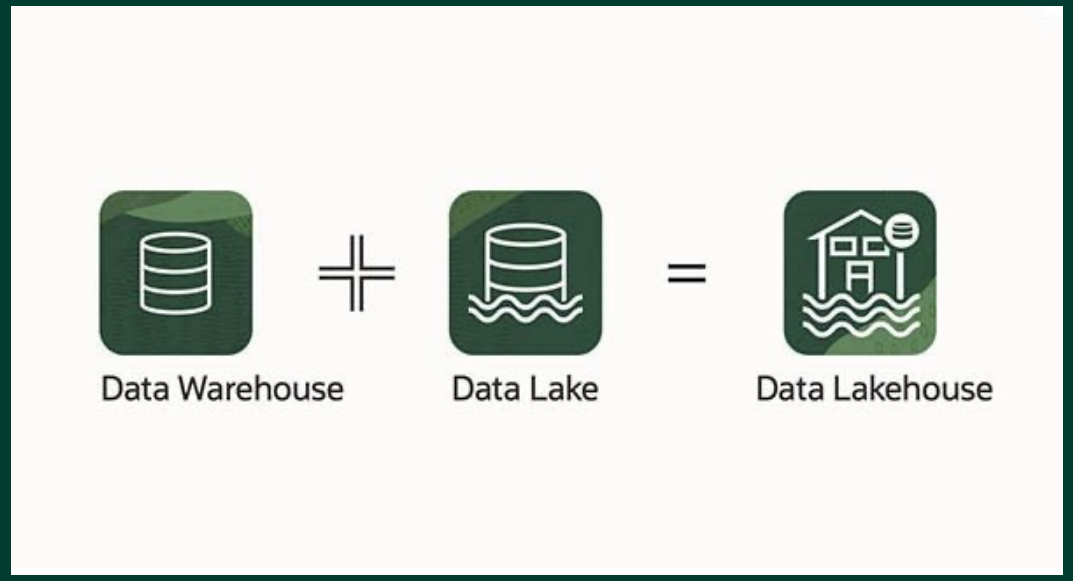


Bitstamp

Towards Data Democratization

Evolving AWS datalake to data lakehouse

Data Platform Engineering

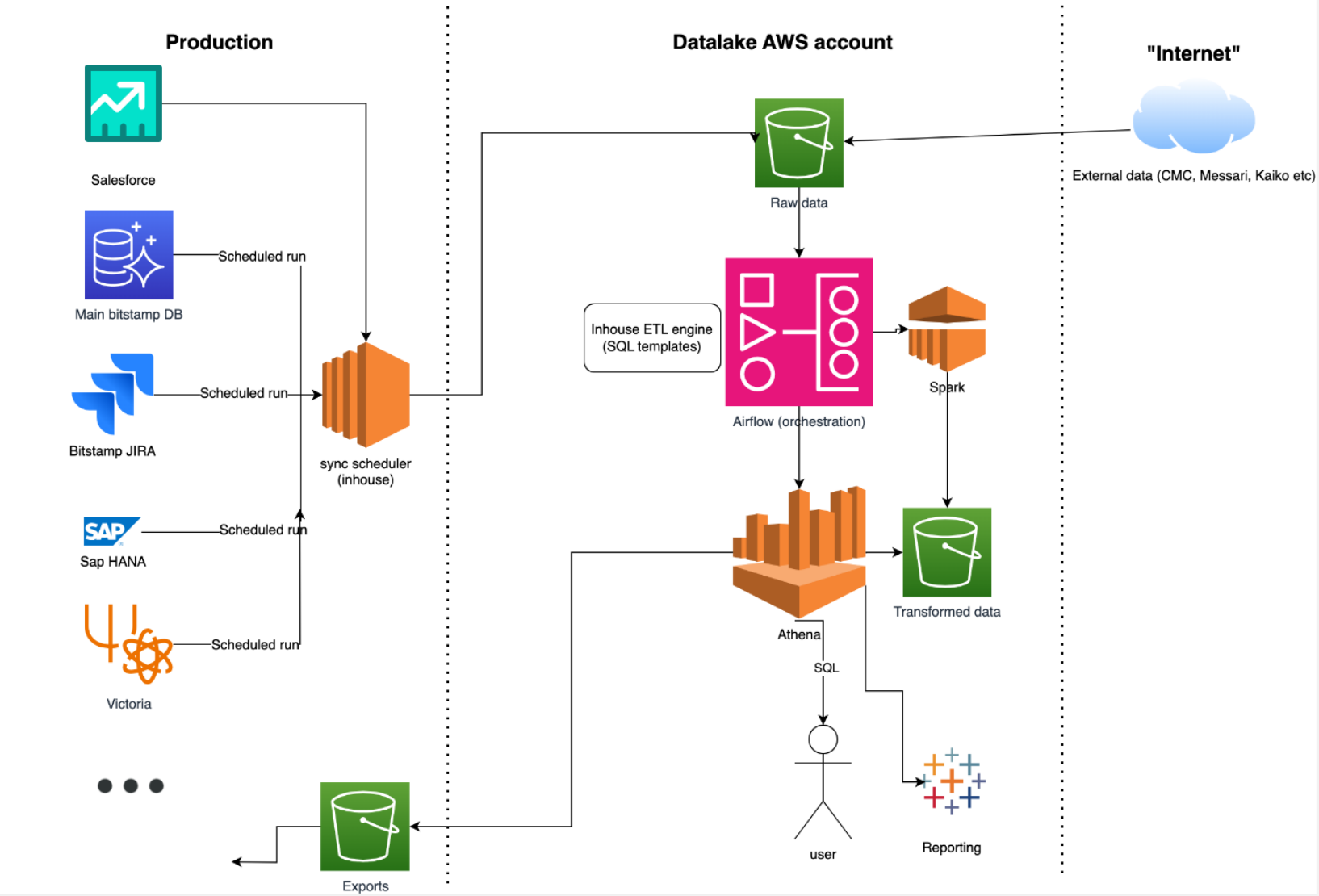
Jake Demšar, April 2024

Our datalake and its problem

A short story

1. We conceptualise our **datalake**
2. Its slowly but surely degenerates into a data **swamp**
3. The **pain** is not great enough
4. The datalake **grows**, the userbase **grows**, the demands **grow**
5. The **pain is great enough**
6. We salvage the situation by introducing data **lakehouse**
7. **< we are here >**
8. The proposed golden paths and pattern sustain a healty data-driven culture

Athena-based datalake

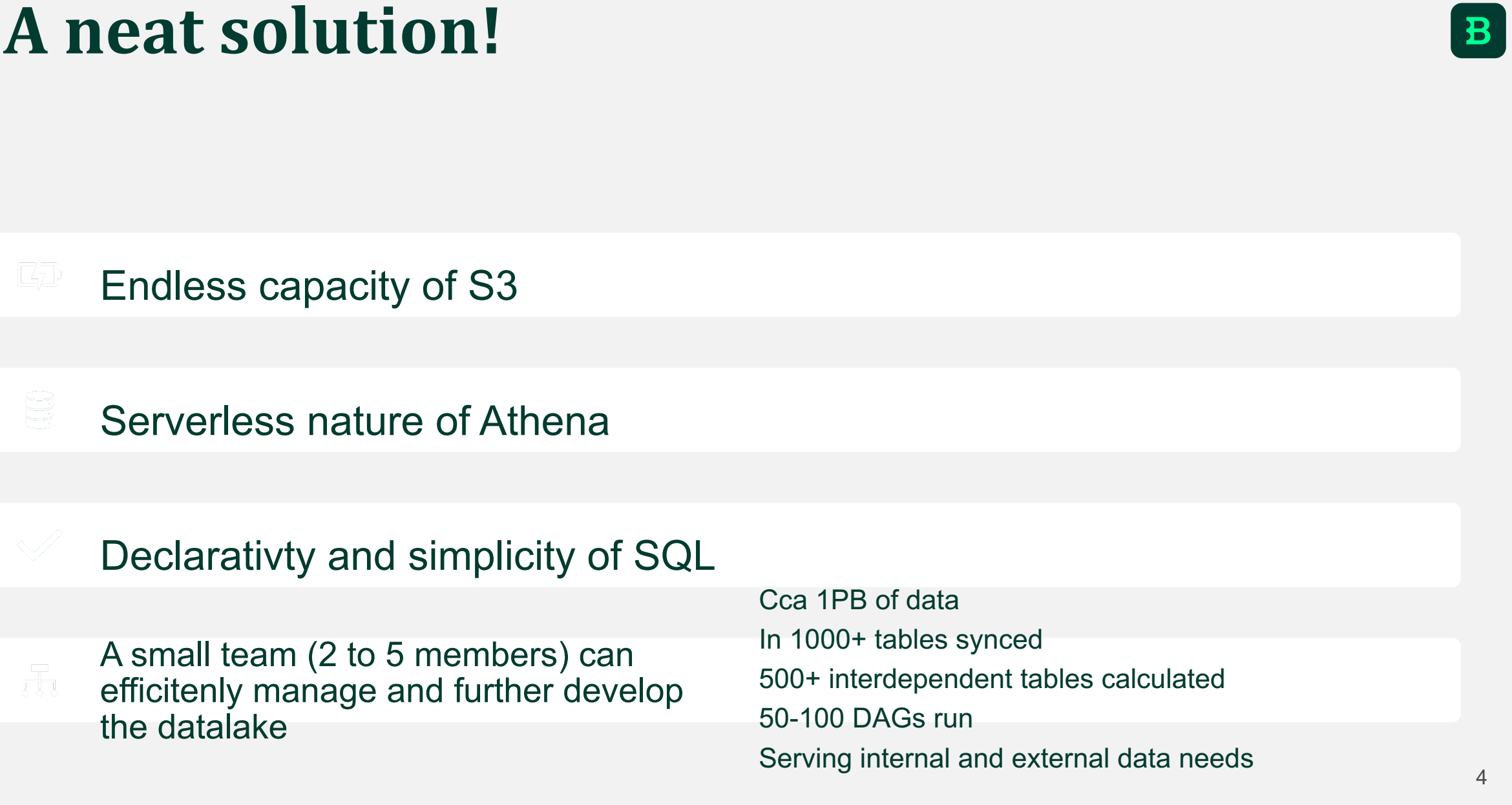


1. Hoard raw data in **S3**
2. Expose raw data in Athena
3. Use **in-house ETL** (SQL templates) + Athena to store transformed data (`.parquet`) in S3
4. Expose `parquets` via **Glue** catalog (“thin layer”)
5. Serve data to users via **Athena** SQL interface

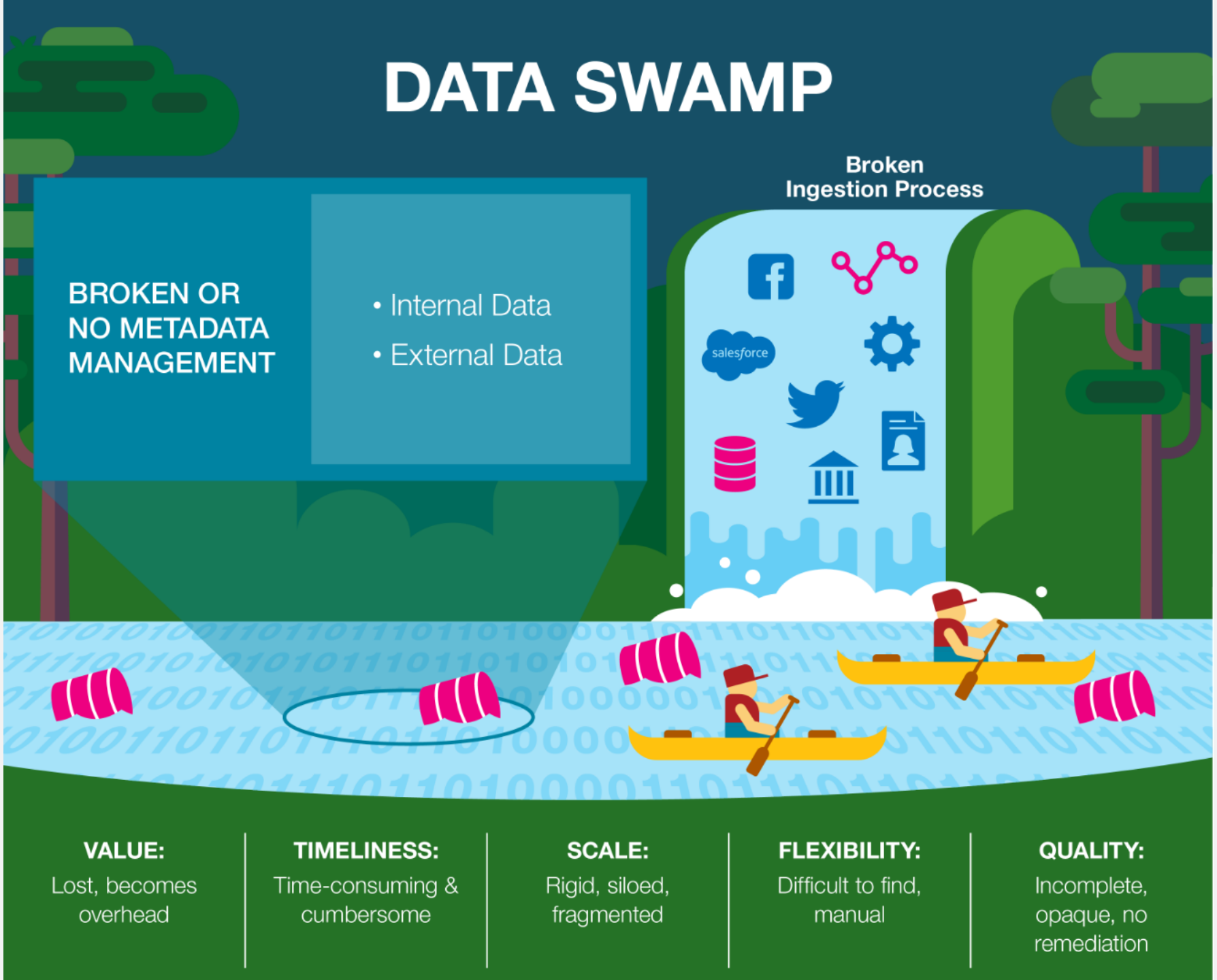
Orchestration: **Airflow**
Data catalog: **Amundsen**

IDEMPOTENT

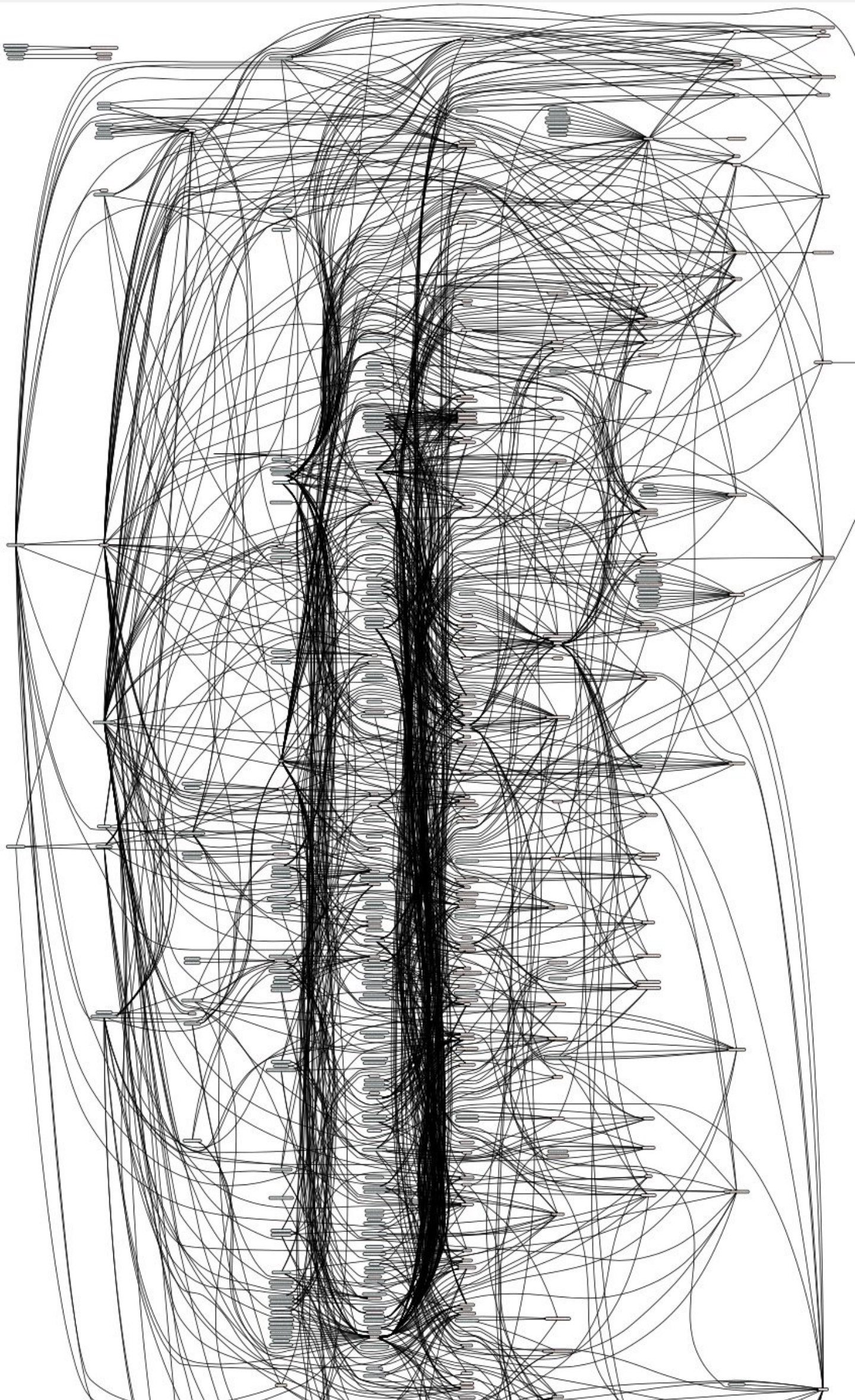
A neat solution!

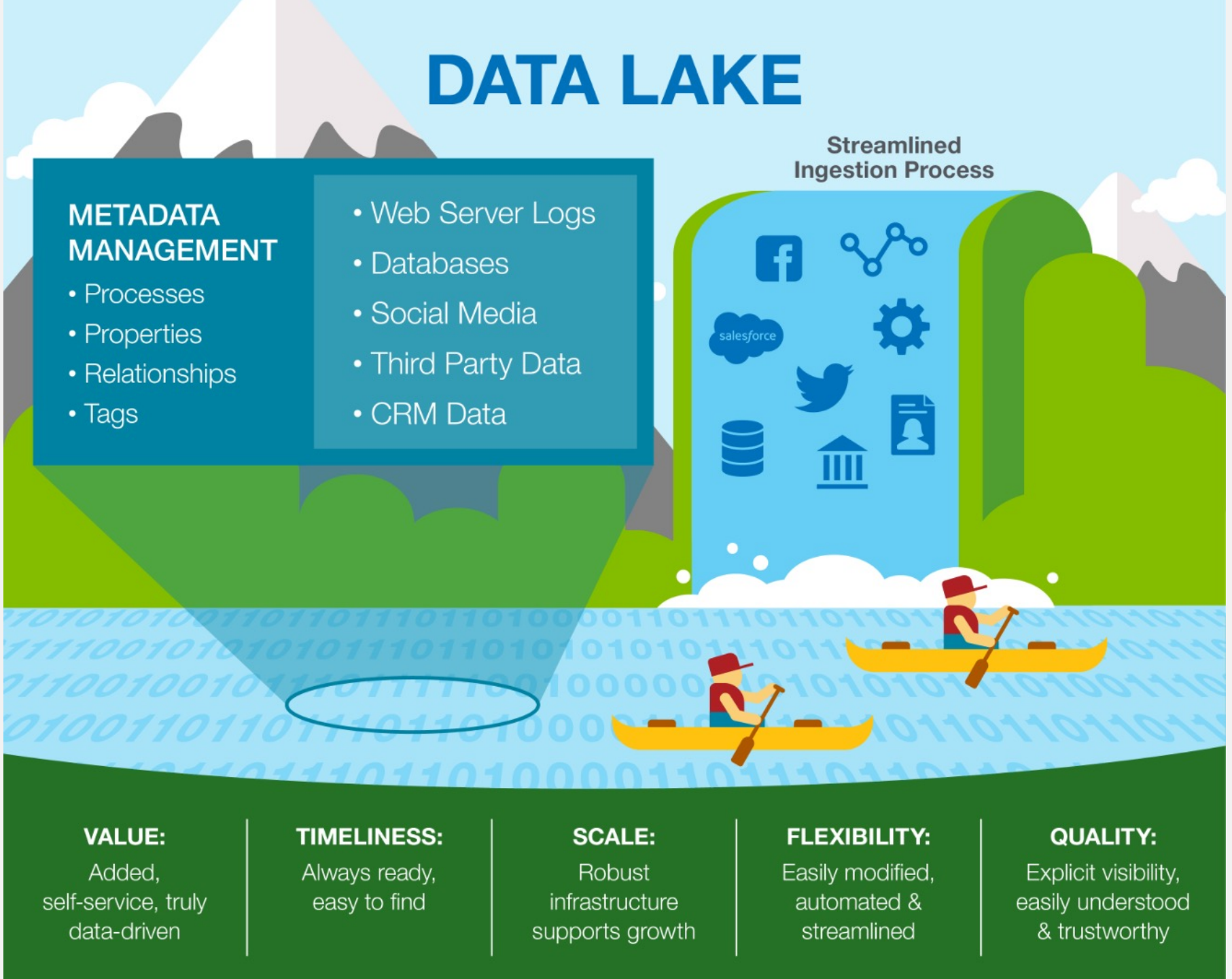


What is the problem, then?



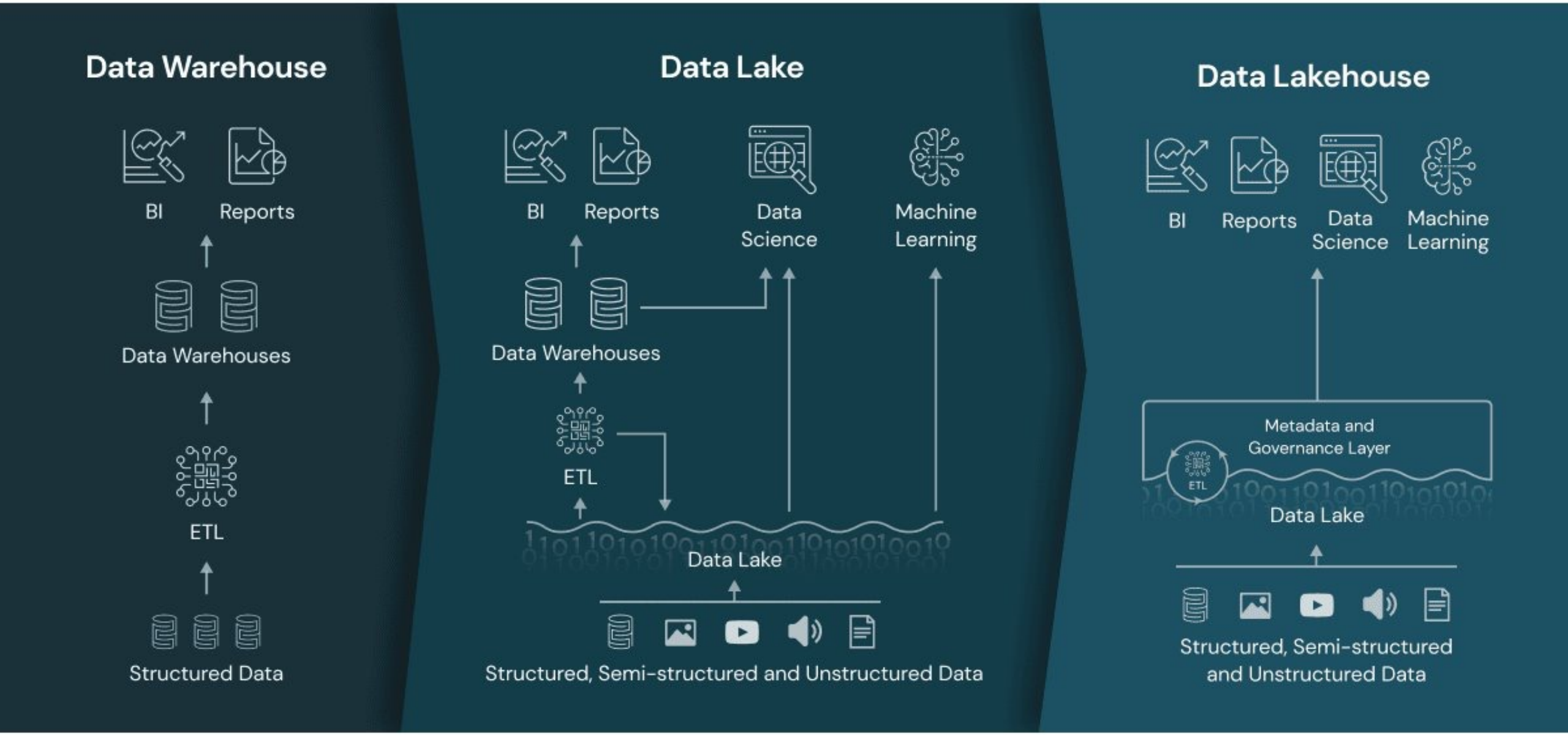
Data ~~mesh~~ mess




DATA LAKE
Streamlined
Ingestion Process
METADATA
MANAGEMENT
• Processes
• Properties
• Relationships
• Tags
• Web Server Logs
• Databases
• Social Media
• Third Party Data
• CRM Data
salesforce
VALUE:
Added, self-service, truly data-driven
TIMELINESS:
Always ready, easy to find
SCALE:
Robust infrastructure supports growth
FLEXIBILITY:
Easily modified, automated & streamlined
QUALITY:
Explicit visibility, easily understood & trustworthy

What we have

What we want



Why lakehouse?

	Data lake	Data lakehouse
Data format	Open format	Open format
Types of data	All types: Structured data, semi-structured data, textual data, unstructured (raw) data	All types: Structured data, semi-structured data, textual data, unstructured (raw) data
Data access	Open APIs for direct access to files with SQL, R, Python and other languages	Open APIs for direct access to files with SQL, R, Python and other languages
Reliability	Low quality, data swamp	**High quality, reliable data with ACID transactions**
Governance and security	Poor governance as security needs to be applied to files	**Fine-grained security and governance for row/columnar level for tables**
Performance	Low	**High**
Scalability	Scales to hold any amount of data at low cost, regardless of type	**Scales to hold any amount of data at low cost, regardless of type**
Use case support	Limited to machine learning	**One data architecture** for BI, SQL and machine learning
Total cost of ownership	Low	Low

How?

Organisational changes

- **Shift left**
- Use **config-based** approach
- Introduce **end-to-end testing**
- Provide toolkits, **golden paths** and conventions.
- **Devs shouldn't think too much.**
- **Data governance** and ownership are maintained on all layers.

Provide **platform**

Domains in *Layers*

DATA FLOW

SOURCE

Data Providers
- production apps
- enterprise apps
- external sources
- temporary data

monolith

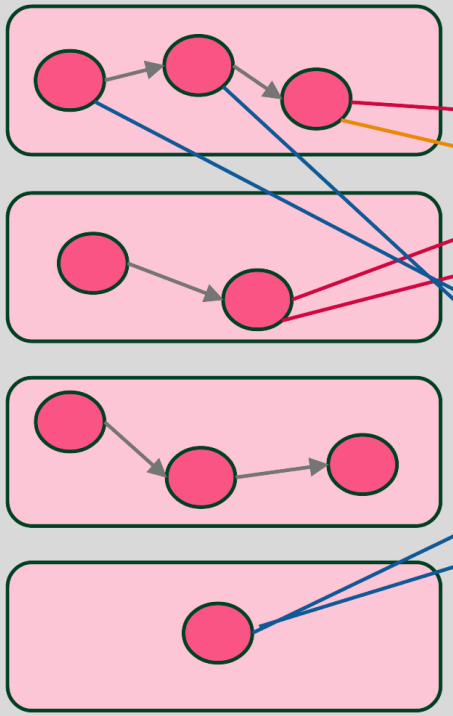
SAP

Kaiko

Atlassian

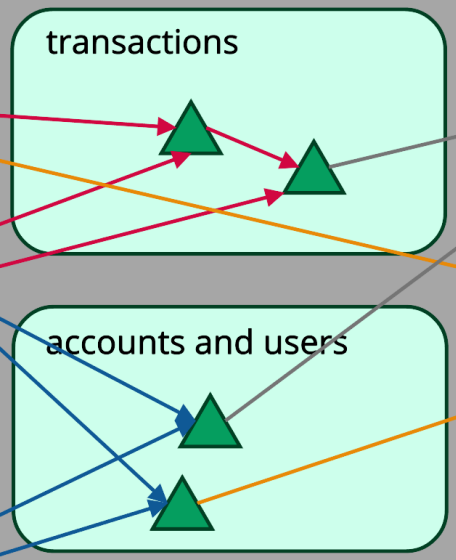
STAGING

Ingested data + base ETLs (closed system)



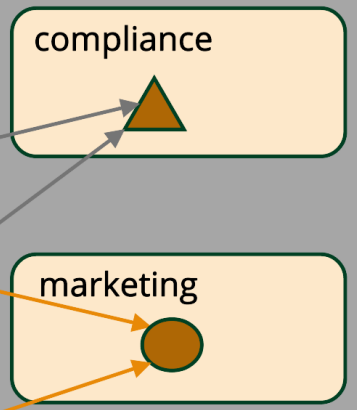
MARTS

Stable schemas



SPECIAL

Mgmt/Highly specialised/experiemental



EXPOSURES

Ad-hoc data stores in business inteligence tools

Ad-hoc data analysis, research, etc.

Tableau, QuickSight, Athena, Excel, SageMaker

EGRESS

- Exports
- Sources being queried by outside applications

Responsibility

Data Integration Owner = Developer (product) = Maintainer (enterprise) = Data Engineer (external)	Data Architect (designs) Data Domain Owners (implement and maintain)	Data Architect (designs) Data Domain Owners (implement and maintain)	Users (anyone)

How?

Technological changes

- Change:
  - AWS Athena → Iceberg
  - Inhouse ETL tooling → DBT (data build tool)
  - Add governance layer → (AWS Data Zone)

Provide **platform**

How?

Technological changes

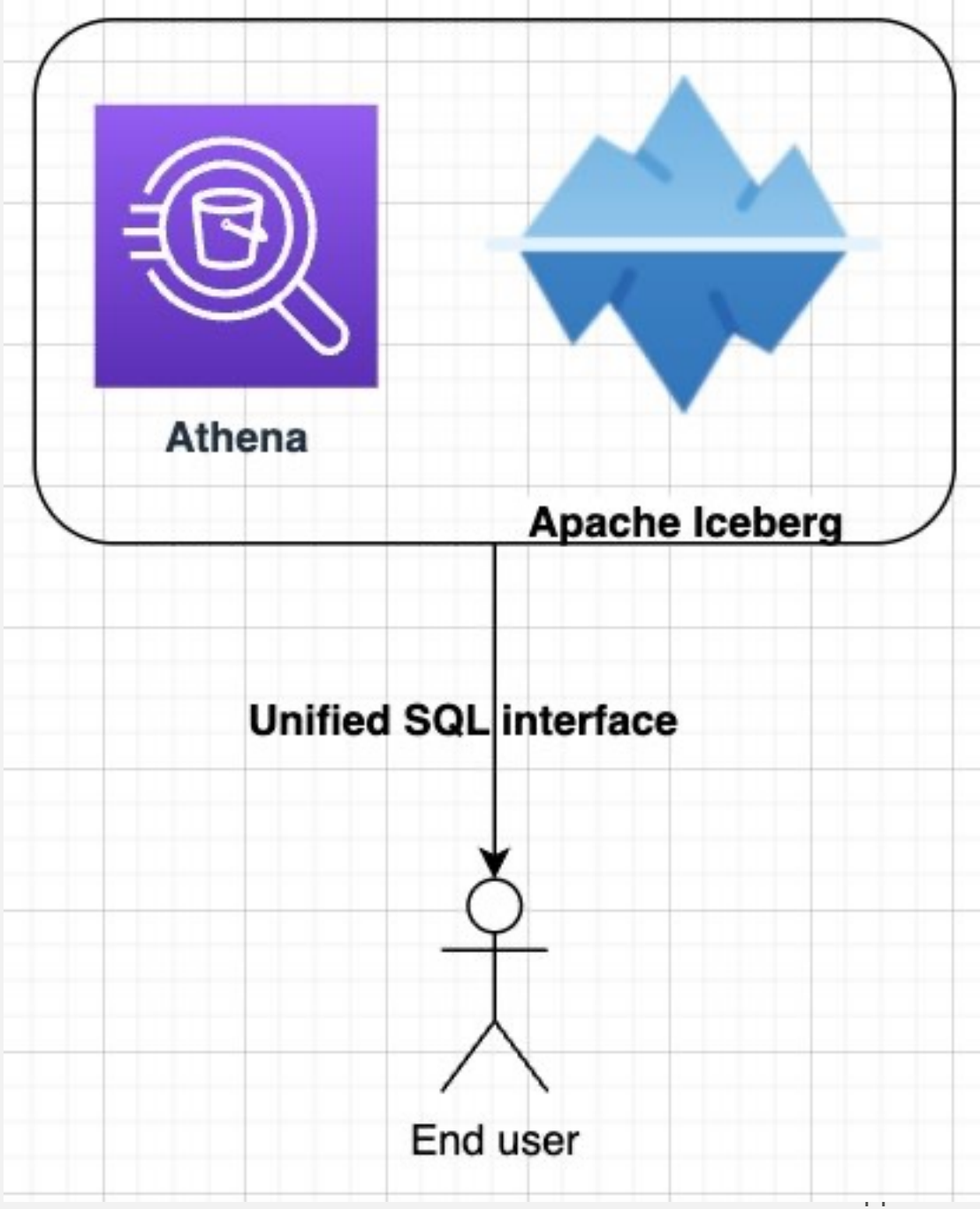
- Change:
  - AWS Athena → **Iceberg**
  - Inhouse ETL tooling → DBT (data build tool)
  - Add governance layer → (AWS Data Zone)

Provide **platform**

From Athena to Iceberg

- ACID transactions
- CRUD operations
- Schema evolution
- Time travel
- Rollback
- Data Compaction
- Compatible with Athena
- Common SQL

TL;DR – **Athena++ for Data Lakehouse**



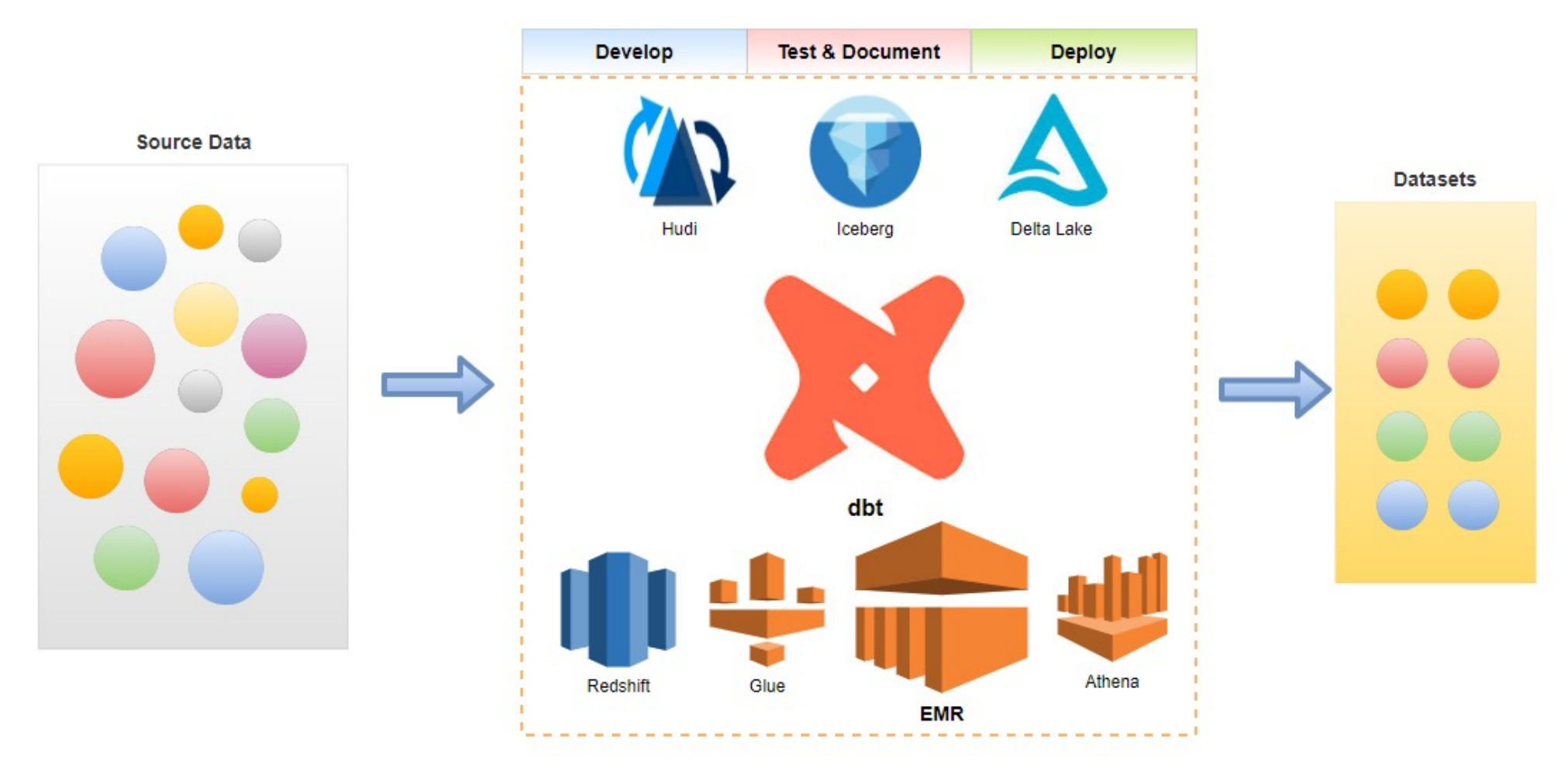
How?

Technological changes

- Change:
  - AWS Athena → Iceberg
  - Inhouse ETL tooling → **DBT (data build tool)**
  - Add governance layer → (AWS Data Zone)

Provide **platform**

DBT - Data Build Tool



Implementing robust ETLs reduces to “writing SQL”

DBT

= SE best practices + data

- **Declarative:** transform user-written code into raw SQL and run it against an engine
- **Git** workflow
- Data **lineage**
- **Testing**
- Support various **materialization** strategies
- In practice: “**SQL and Jinja**”

```
jaffle_shop
├── README.md
├── analyses
├── seeds
│   └── employees.csv
├── dbt_project.yml
├── macros
│   └── cents_to_dollars.sql
├── models
│   ├── intermediate
│   │   └── finance
│   │       ├── _int_finance__models.yml
│   │       └── int_payments_pivoted_to_orders.sql
│   ├── marts
│   │   ├── finance
│   │   │   ├── _finance__models.yml
│   │   │   ├── orders.sql
│   │   │   └── payments.sql
│   │   └── marketing
│   │       ├── _marketing__models.yml
│   │       └── customers.sql
│   ├── staging
│   │   ├── jaffle_shop
│   │   │   ├── _jaffle_shop__docs.md
│   │   │   ├── _jaffle_shop__models.yml
│   │   │   ├── _jaffle_shop__sources.yml
│   │   │   ├── base
│   │   │   │   ├── base_jaffle_shop__customers.sql
│   │   │   │   └── base_jaffle_shop__deleted_customers.sql
│   │   │   ├── stg_jaffle_shop__customers.sql
│   │   │   └── stg_jaffle_shop__orders.sql
│   │   └── stripe
│   │       ├── _stripe__models.yml
│   │       ├── _stripe__sources.yml
│   │       └── stg_stripe__payments.sql
│   └── utilities
│       └── all_dates.sql
├── packages.yml
├── snapshots
└── tests
    └── assert_positive_value_for_total_amount.sql
```

How?

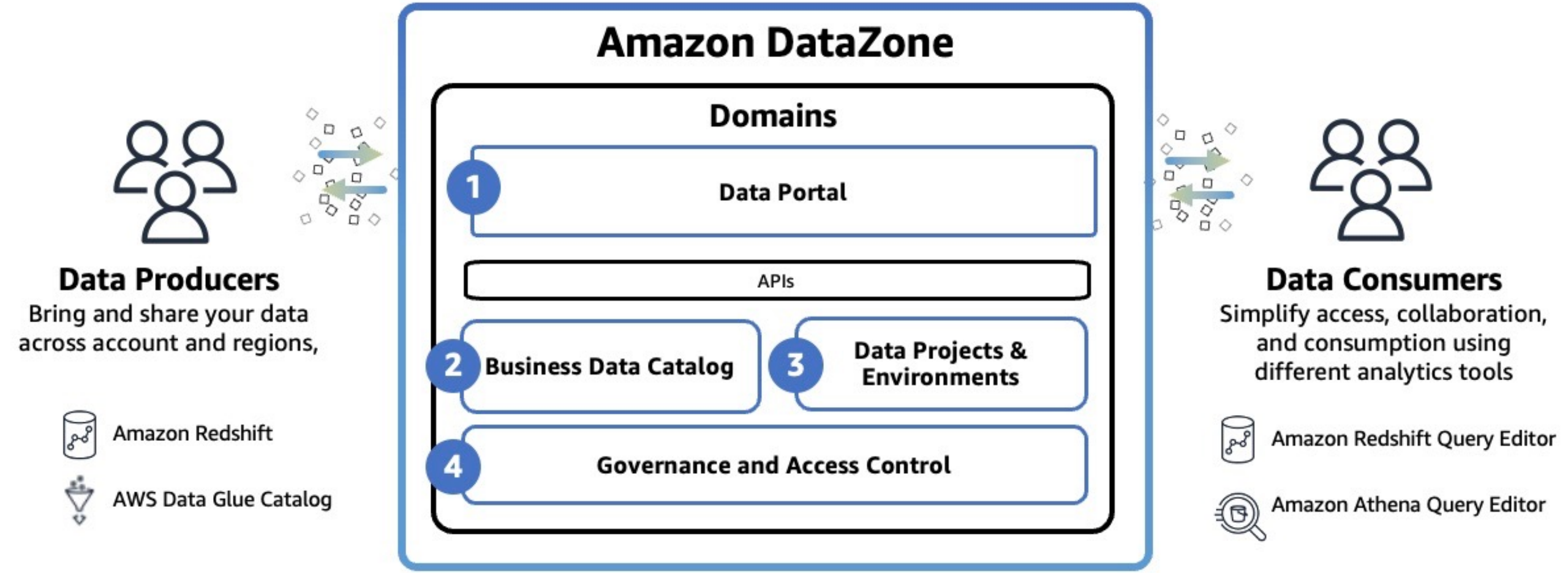
Technological changes

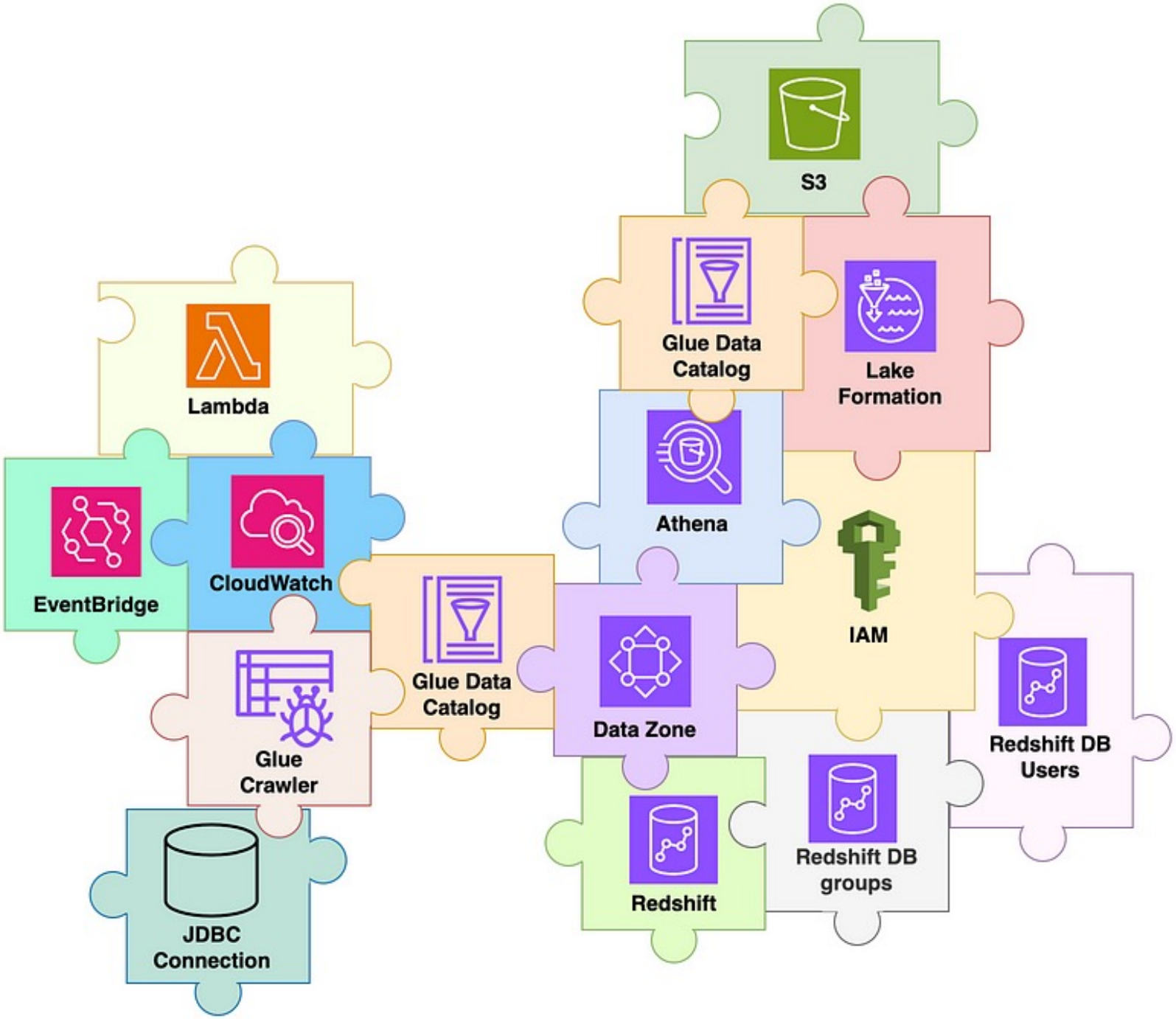
- Change:
  - AWS Athena → Iceberg
  - Inhouse ETL tooling → DBT (data build tool)
  - Add governance layer → (**AWS Data Zone**)

Provide **platform**

AWS Data Zone to the rescue

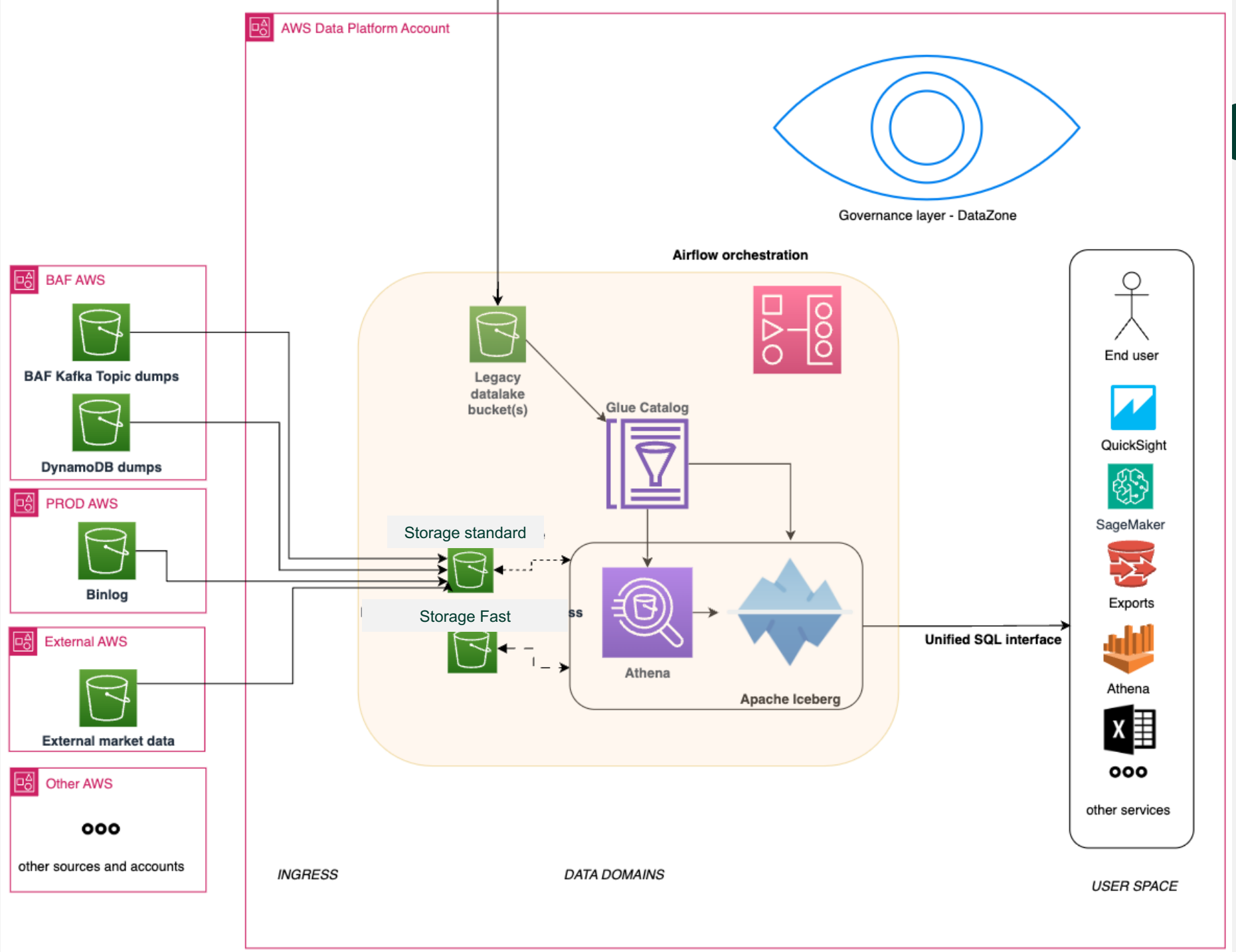
Amazon‘s new offering for data governance




S3
Glue Data Catalog
Lake Formation
Lambda
Athena
EventBridge
CloudWatch
IAM
Glue Data Catalog
Data Zone
Redshift DB Users
Glue Crawler
Redshift
Redshift DB groups
JDBC Connection

Lakehouse Infrastructure

- Shift ingress left
- Apache Iceberg-
- All-seeing governance
- Liberated user space



Highlights

- **Data swamp is salvageable**
- We **embraced legacy** data, we do not fight it (too much)
- We were **lucky** to have a sort of lakehouse already set up
- **Iceberg** is neat, **DBT** is neat
- **Golden paths**, conventions and governance is the name of the game

Unchecked, at some point, every datalake will become “too big” for a single team

Challanges

- **Shifting left** and scaling
  - Technologically
  - Culturally
- Onboarding to **DBT**
  - Introducing software practices into traditionally non-software teams
- Adapting AWS **DataZone**
  - New product, pilot customers

Q & A

Bitstamp