# Self-serve data ingestion for high-scale workloads in the cloud

upsolver

# Upsolver

## What we do

Upsolver is a self-serve cloud data ingestion service for high-scale workloads *(big data, streaming, AI)*.

Product is purpose-built for the cloud with a Snowflake-like architecture for data processing.

## Certifications

SOC 2 TYPE II CERTIFIED

aws Qualified Software

## Partners

aws partner network
Advanced Technology Partner
Data & Analytics Competency

snowflake®

dbt

## Proven at scale

| | |
|---|---|
| 8% of customers | >1PB/mon |
| 26% of customers | >100TB/mon |
| 54% of customers | >10TB/mon |
| 82% of customers | >1TB/mon |

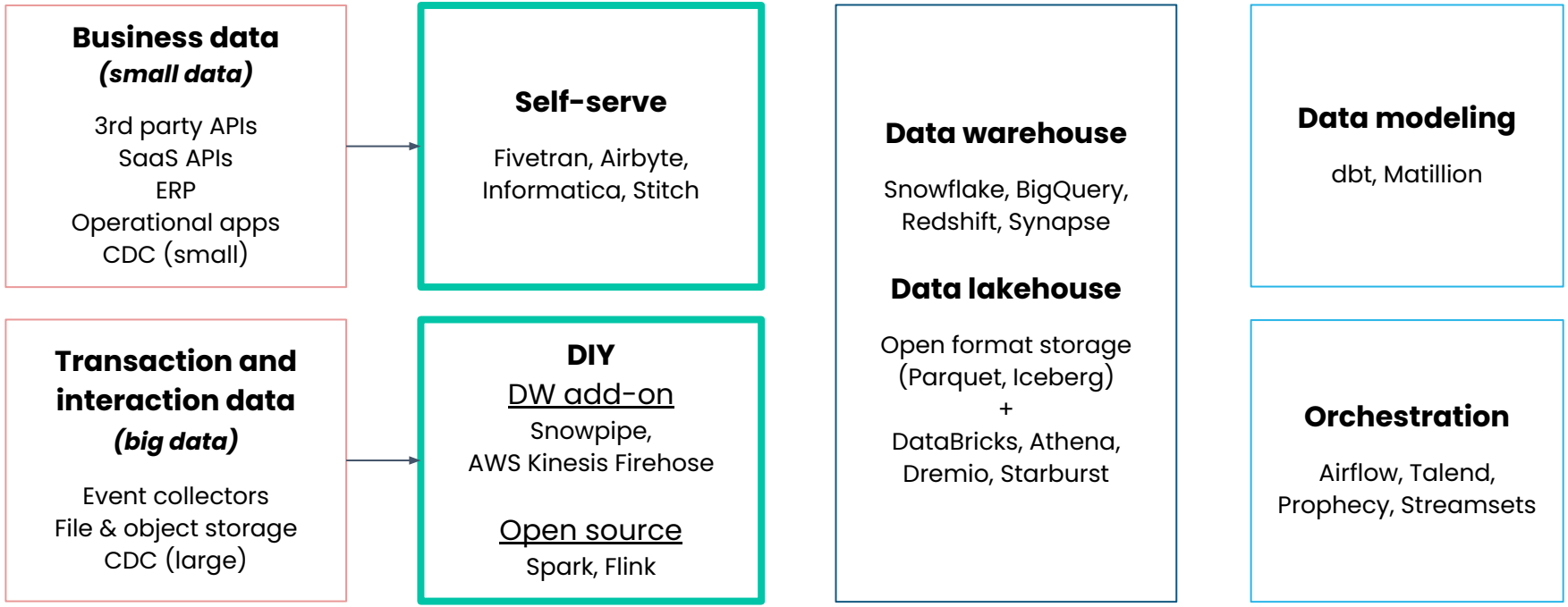
# The gap

1. Market map
2. Alternatives for high-scale ingestion
3. The tooling gap for high-scale ingestion

upsolver

# Ingestion in the modern data stack

The modern data stack (MDS) makes it simple for companies to become data-driven





# For high-scale, existing tools fall short

## Ingestion

**Self-serve**

Fivetran, Airbyte,
Informatica, Stitch

**DIY**

DW add-on
Snowpipe,
AWS Kinesis Firehose

Open source
Spark, Flink

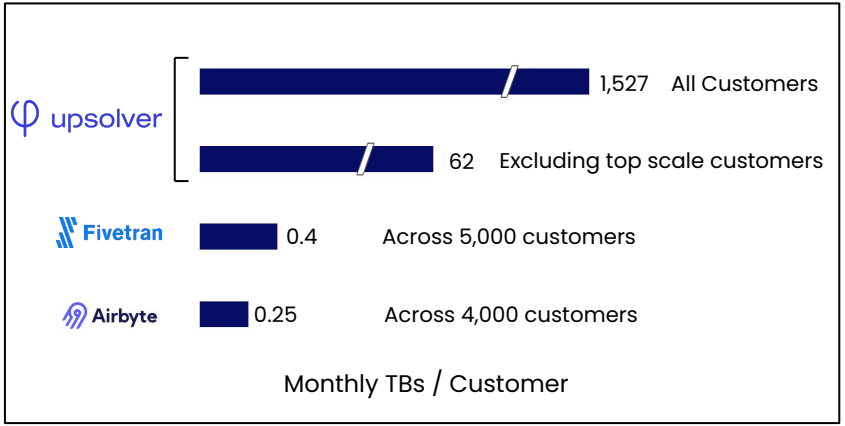
**Challenge**
NOT built or priced for high-scale

**Challenge**
Data engineering bottleneck limits agility and causes many data quality incidents

# Self-serve ingestion tools are NOT built/priced for scale

Product focus is long tail connectors (SaaS, small CDC) with small business data

## Today, customers use self-serve tools to process small scale data




Fun fact

Upsolver's 2nd biggest customer has more data than all 9,000 Fivetran & Airbyte customers combined

## At scale, self-serve tools' cost per month is 10X+ higher than Upsolver & DIY

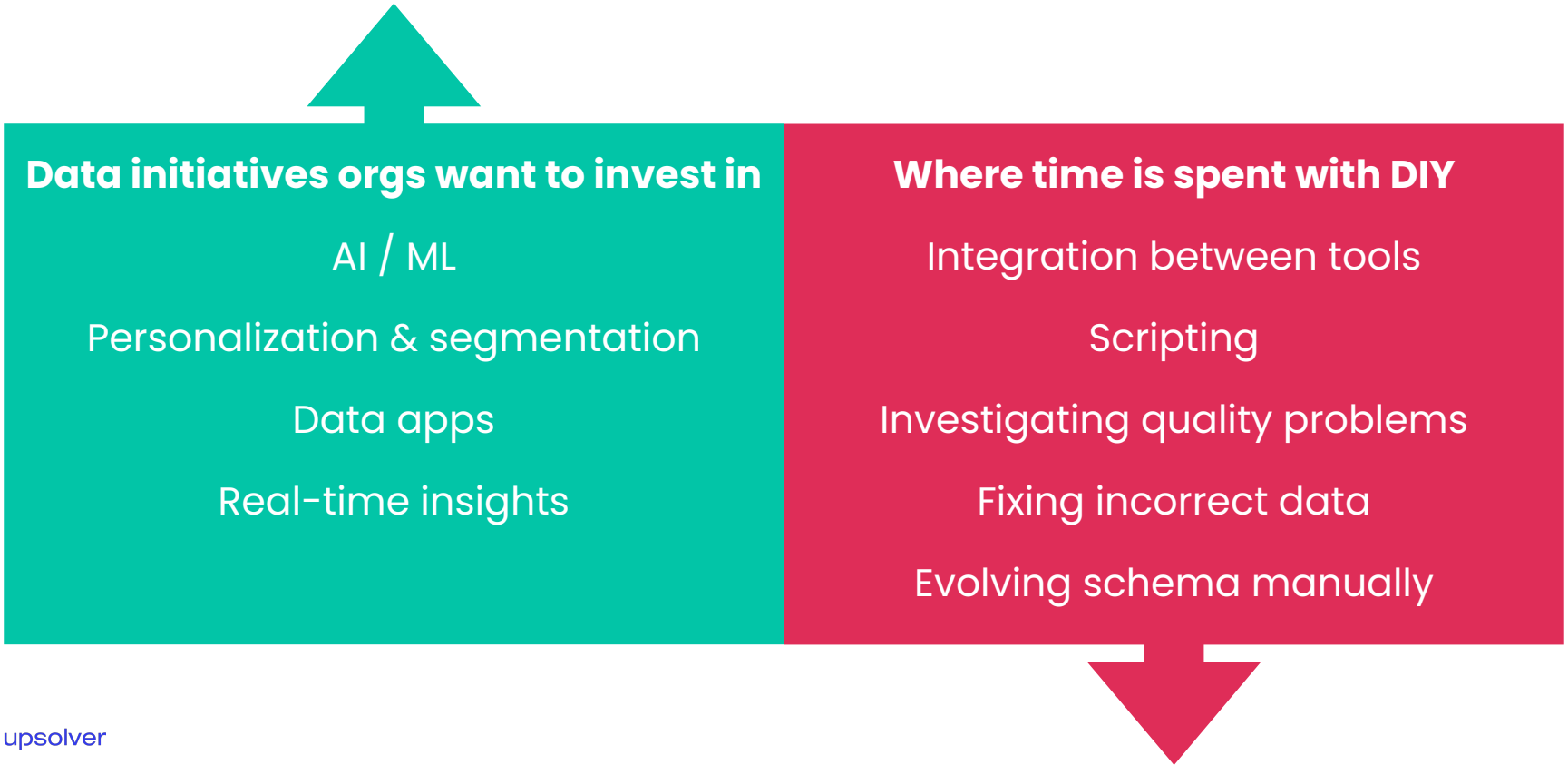
| Scale | Fivetran | Airbyte | Upsolver |
|---|---|---|---|
| 1 TBs | $22,000 | $5,000 | $4,225 |
| 10 TBs | $50,000 | $20,000 | $6,250 |
| 100 TBs | $500,000 | $200,000 | $26,500 |

*Assumptions:*

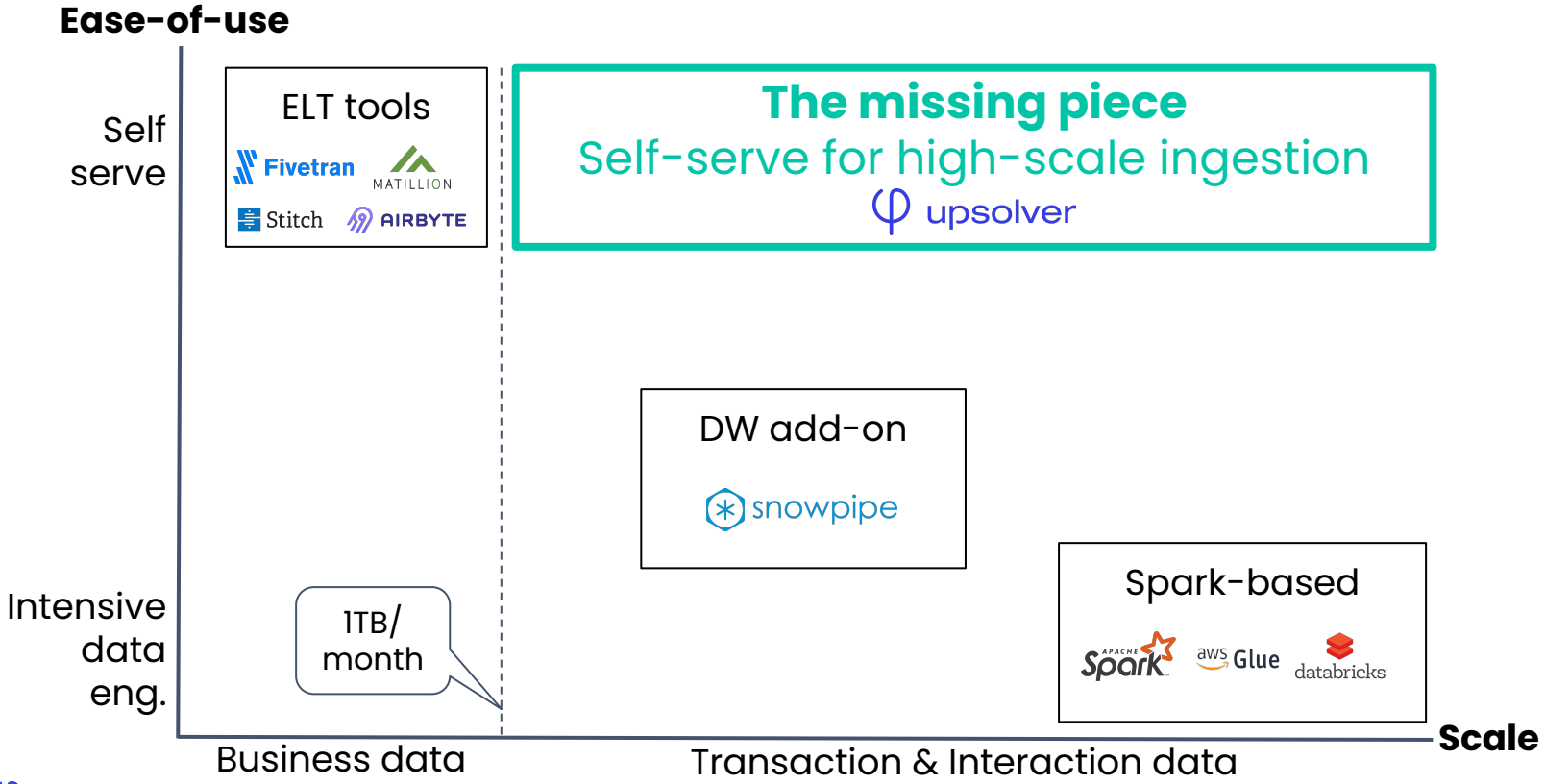
- *Append-only ingestion, 1KB per row*
- *Upsolver list price - $4,000 + #TBs*$225*

# DIY is a bottleneck to becoming data-driven

Counter to MDS ethos of pursuing higher value activities, not DIY data engineering



# Upsolver fills the tooling gap for high-scale ingestion



# Product

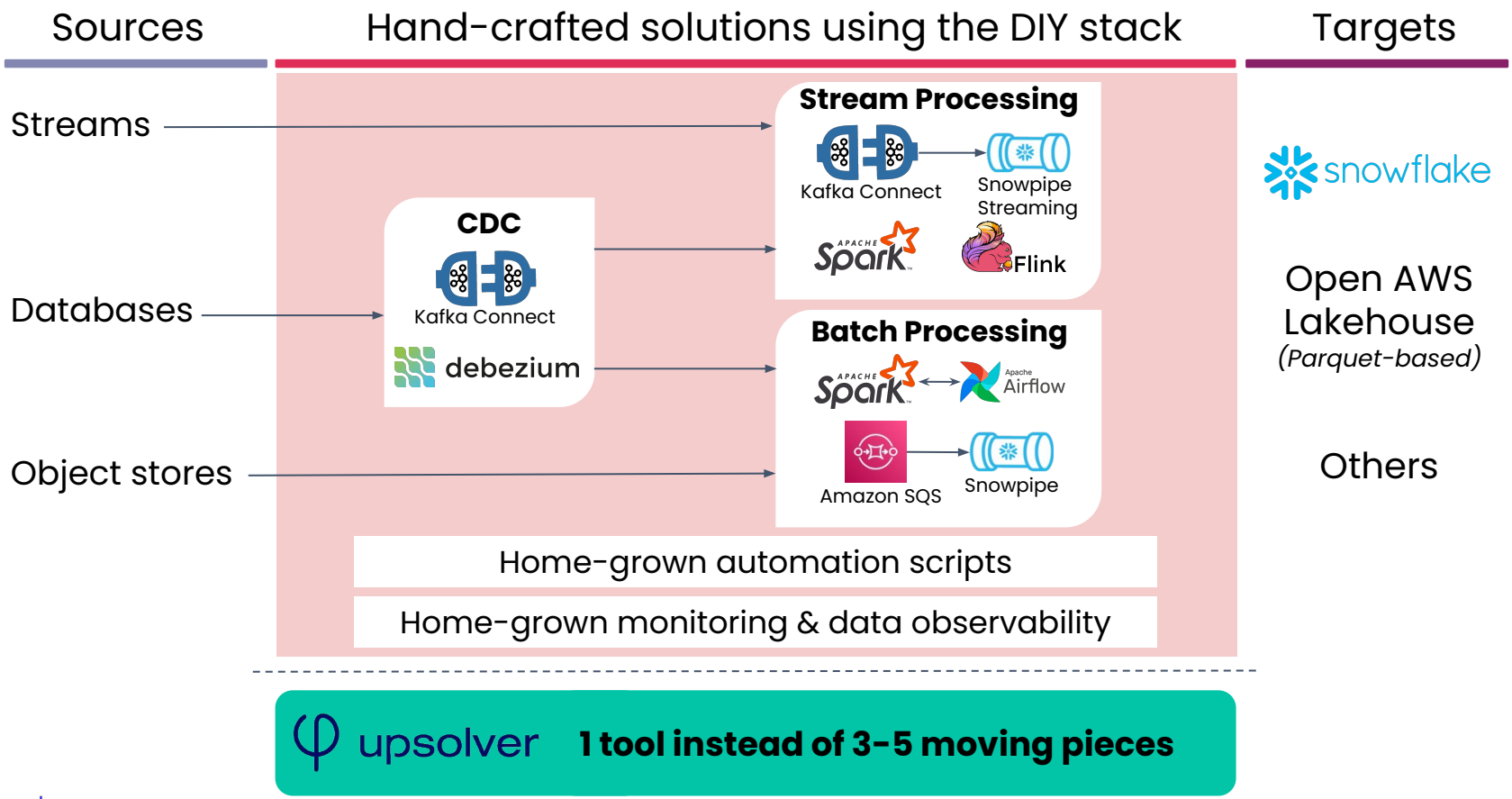
- The capabilities that solve the DIY complexity
  1. ETL using a single tool
  2. No-code & low-code dev experience
  3. Automatic prevention of most data disruptions
  4. Controlling quality - discover, resolve, prevent

upsolver

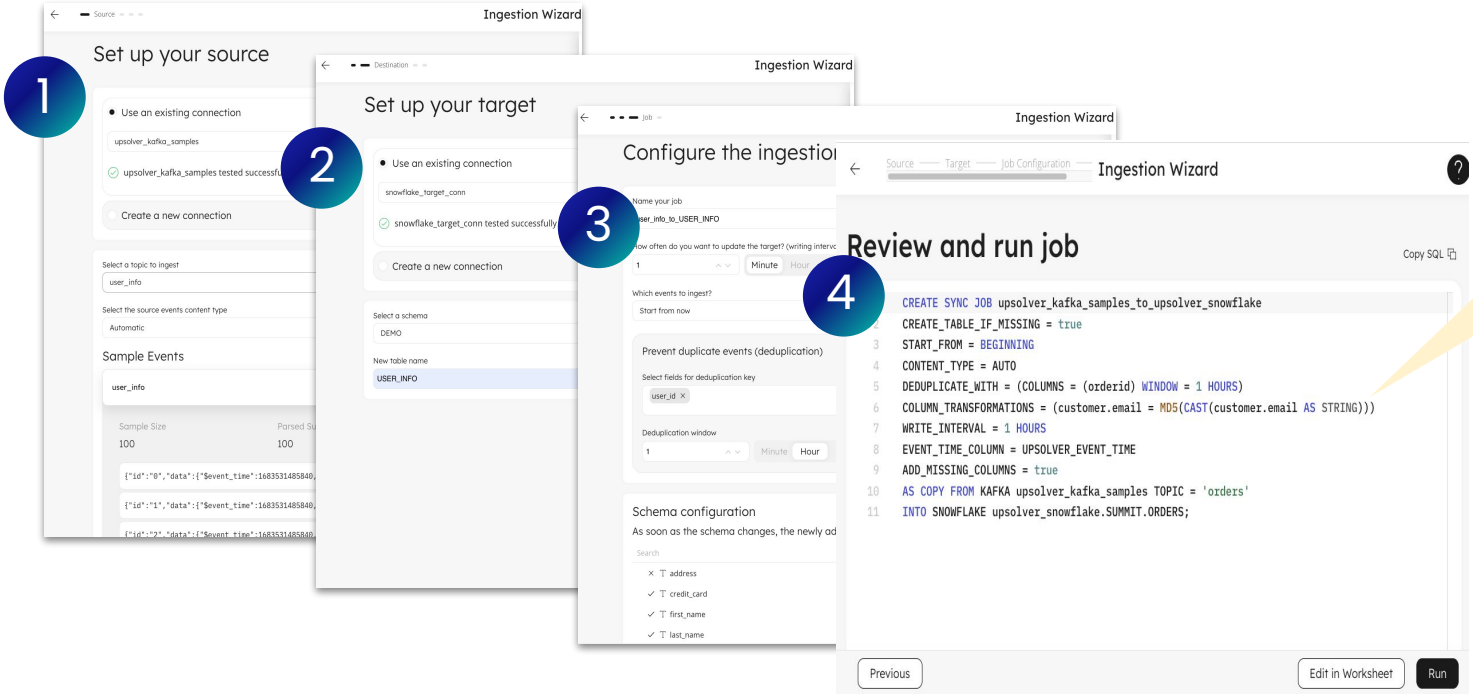
# Upsolver uniquely bridges the DIY complexity gap

| | DIY - Open source (Spark-based) | DIY - DW add-on (Snowpipe) | Self-serve (Upsolver) |
|---|---|---|---|
| Single tool for ETL | No | No | Yes |
| No-code and low-code dev experience | No | No | Yes |
| Automatically prevention of most data disruptions | No | No | Yes |
| Built-in tools for discovering, resolving and preventing data quality issues | No | No | Yes |
| **Outcome** | Months to implement, Constant break-fix and ops headaches | | Hours to implement, near-zero ops |

# (1) ETL using a single tool



# (2) No-code & low-code dev experiences




Code is simple and therefore easy to understand, test and manage

## Dev options

Upsolver GUI

Upsolver Worksheet

dbt

CLI

SDK

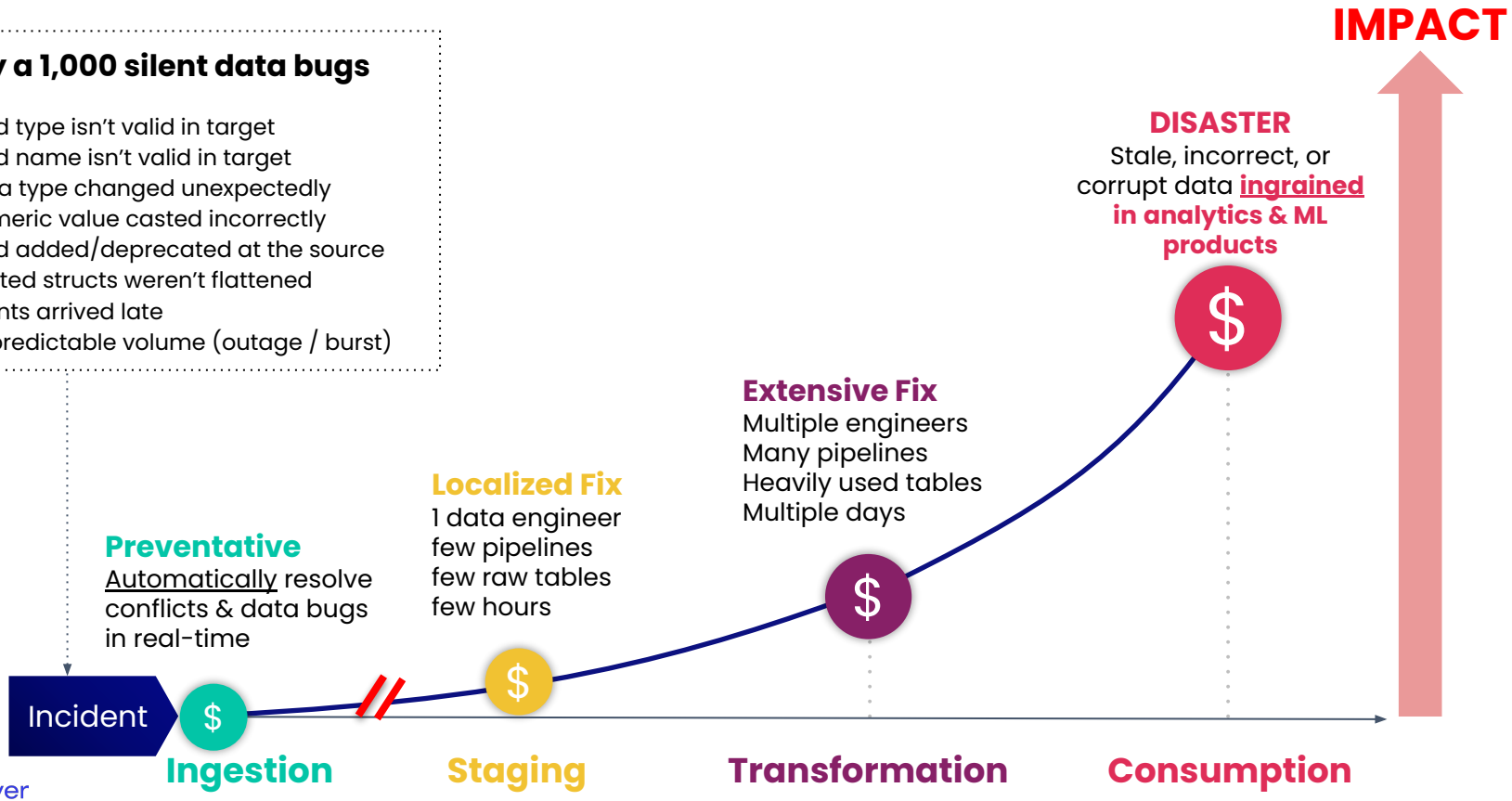
| load_user_events | • Running | Up to date | 0 | | 14 days ago | |
|---|---|---|---|---|---|---|

upsolver

# (3) Automatic prevention of most data disruptions

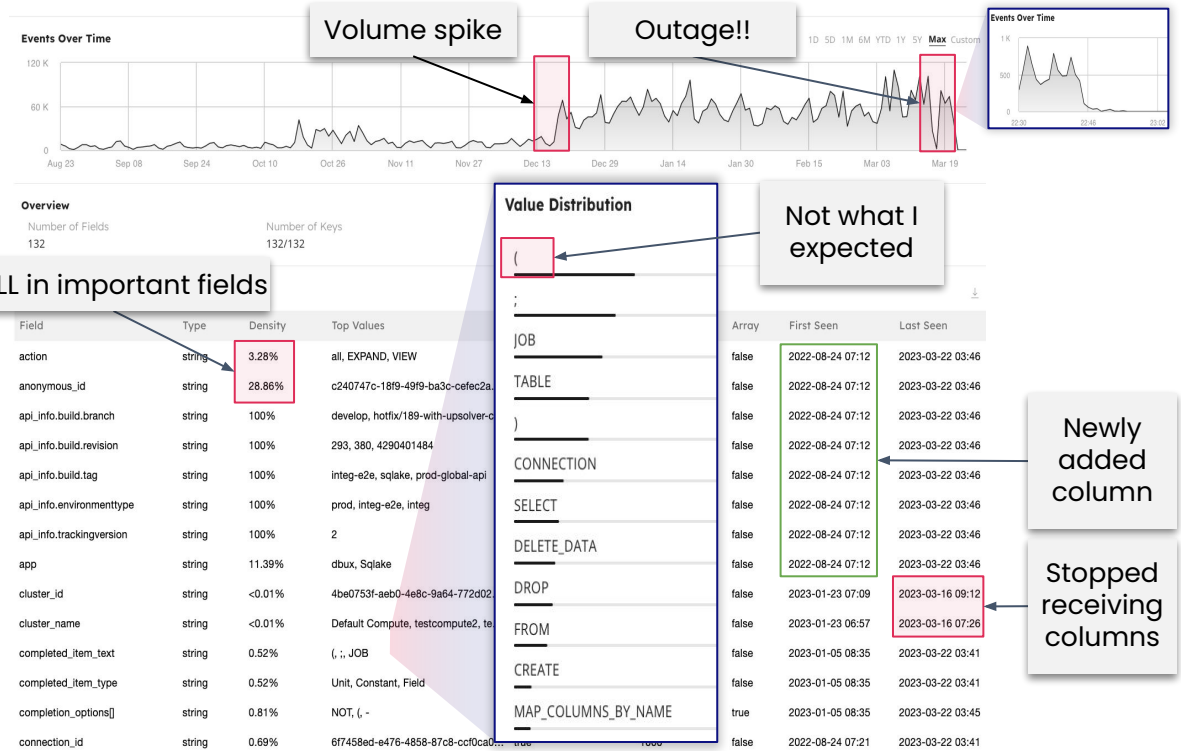
**Death by a 1,000 silent data bugs**

- Field type isn't valid in target
- Field name isn't valid in target
- Data type changed unexpectedly
- Numeric value casted incorrectly
- Field added/deprecated at the source
- Nested structs weren't flattened
- Events arrived late
- Unpredictable volume (outage / burst)



# (4) Controlling quality - discover, resolve, prevent

## Discover using built-in data observability




## Resolve & Prevent

```
CREATE SYNC JOB ingest_from_kafka_to_snowflake
    CONTENT_TYPE = JSON
    START_FROM = BEGINNING
    CREATE_TABLE_IF_MISSING = TRUE
    WRITE_INTERVAL = 1 MINUTE
    DEDUPLICATE_WITH = (COLUMNS = (orderid) WINDOW = 1 MINUTES)
    COLUMN_TRANSFORMATIONS = (customer.email = MD5(customer.email))
AS COPY FROM KAFKA upsolver_kafka TOPIC = 'orders'
INTO SNOWFLAKE upsolver_snowflake.production.ORDERS_DATA
WITH EXPECTATION nonzero_nettotal
    CHECK nettotal <> 0 ON VIOLATION WARN
WITH EXPECTATION orderid_notnull
    CHECK orderid IS NOT NULL ON VIOLATION DROP
WITH EXPECTATION invalid_states
    CHECK LENGTH(customer.address.state) = 2 ON VIOLATION WARN;
```

Resolve incorrect data using replay

Prevent incorrect data from entering the warehouse using quality expectations

# Case studies

- VS self-serve ELT
- VS DIY add-on
- VS DIY open source

upsolver

# Technology

1. Cloud-native architecture for high-scale processing
2. Key-value store for big time-series data
3. Intelligent landing zone for raw data
4. Easy-to-use CDC service, built on Apache Debezium

upsolver