

# Data movement for data developers

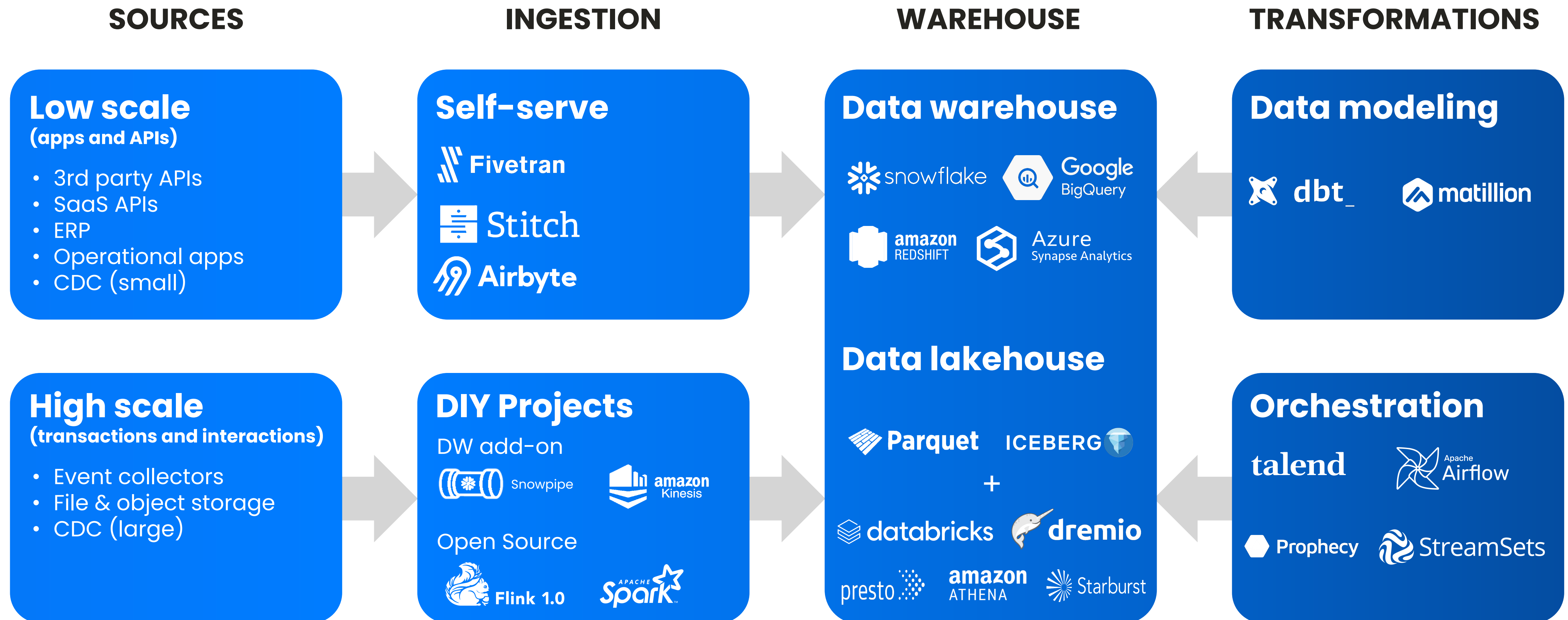
Deliver data generated by your prod environment  
to downstream users at scale, without sacrificing quality



$\varphi$  upsolver

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

# Ingestion is the only category in the modern data stack in which tools vary according to scale



# Self-serve tools do not scale efficiently to production transaction and interaction data

## TERABYTES INGESTED PER MONTH



**0.25**  
4,000 customers



**0.4**  
5,000 customers



**1,537**  
Excluding top scale customers

**1,537**  
All customers

## App & API data is small

Self-serve tools today touch only a **tiny** fraction of customers' data

## But not transaction & interaction data

0.4TB is less data than what 1 million Candy Crush players generate in an hour

## It's a different playing field

Upsolver processes **17X** more data for its **single** largest customer than **all** of Fivetran and Airbyte's 9000 customers **combined**

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

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 projects can handle high-scale but...



**Each analytics cycle takes weeks-months**



**There is a lot of breakage from human errors and bugs**



**Non data engineers don't have access**

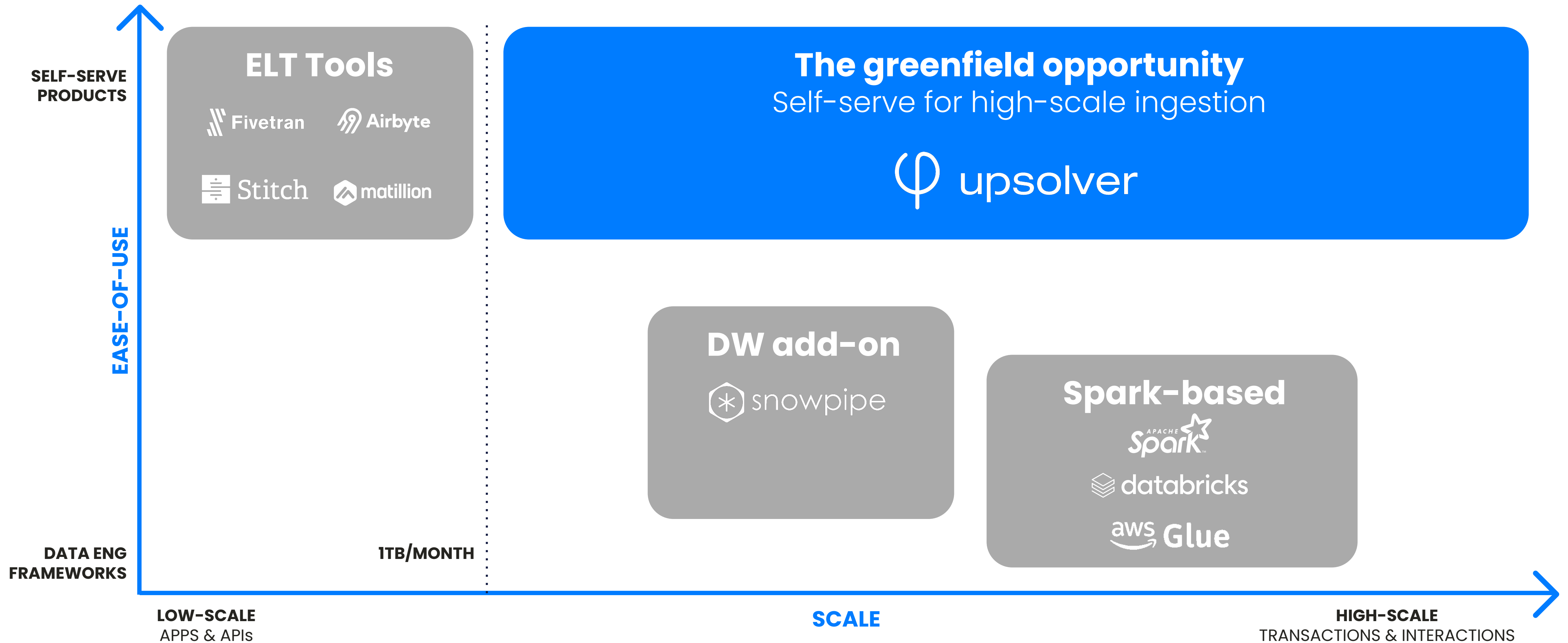


**Data engineering teams don't have attention for new initiatives**

The DIY approach goes against the Modern Data Stack ethos, to use tools that empower self-serve and free up data engineers to focus on high-value activities



# High-scale ingestion must adopt self-serve tools



# $\Psi$ upsolver

Superpowers to overcome high-scale ingestion complexity with self-serve



## Single tool

Single tool for E(T)L instead of many moving pieces



## Coding optional

Coding is optional (no and low code experience)



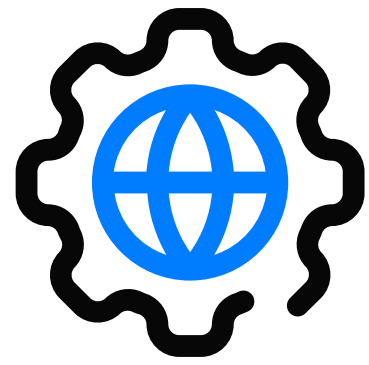
## Quality is built-in

Quality is built-in: discover, resolve, prevent



## Automatic data

Automatic data and schema correction



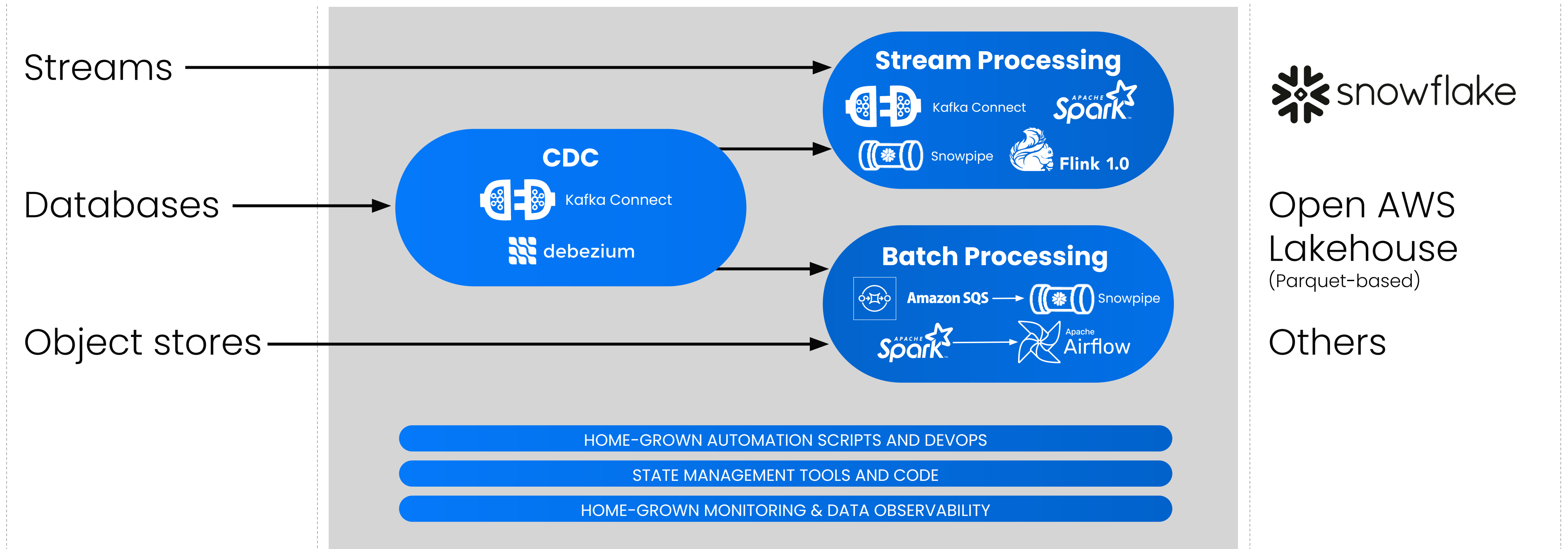
# Single tool

Single tool for E(T)L instead of many moving pieces

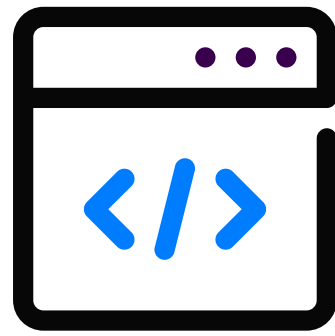
## SOURCES

## HAND-CRAFTED SOLUTIONS USING THE DIY STACK

## TARGETS





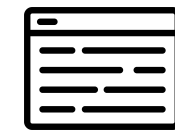


# Coding optional

Coding is optional (no and low code experience)



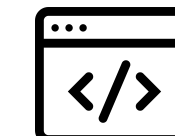
Upsolver GUI



Upsolver Worksheet



dbt



CLI



SDK

No-code ingestion wizards

Simple code which is easy to understand, test and manage in CI/CD processes

The image shows three overlapping screenshots of the Upsolver Ingestion Wizard interface, numbered 1, 2, and 3. Step 1 is 'Set up your source', step 2 is 'Set up your target', and step 3 is 'Configure the ingestion job'. The 'Configure the ingestion job' screen shows fields for job name, update interval, events to ingest, deduplication settings, and schema configuration.



The image shows the 'Review and run job' screen of the Upsolver Ingestion Wizard. It displays the following SQL code:

```

1 CREATE SYNC JOB upsolver_kafka_samples_to_upsolver_snowflake
2 CREATE_TABLE_IF_MISSING = true
3 START_FROM = BEGINNING
4 CONTENT_TYPE = AUTO
5 DEDUPLICATE_WITH = (COLUMNS = (orderid) WINDOW = 1 HOURS)
6 COLUMN_TRANSFORMATIONS = (customer.email = MD5(CAST(customer.email AS STRING)))
7 WRITE_INTERVAL = 1 HOURS
8 EVENT_TIME_COLUMN = UPSOLVER_EVENT_TIME
9 ADD_MISSING_COLUMNS = true
10 AS COPY FROM KAFKA upsolver_kafka_samples TOPIC = 'orders'
11 INTO SNOWFLAKE upsolver_snowflake.SUMMIT.ORDERS;

```

Buttons for 'Previous', 'Edit in Worksheet', and 'Run' are visible at the bottom.

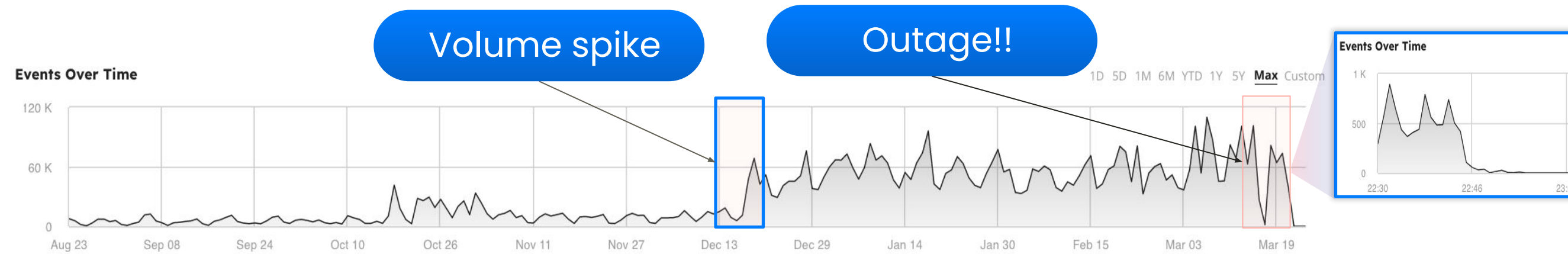




# Quality is built-in: discover, resolve, prevent

DIY projects require hand-crafted observability and quality

## DISCOVER USING BUILT-IN DATA OBSERVABILITY



### Overview

Number of Fields: 132  
Number of Keys: 132/132

NULL in important fields

Field	Type	Density	Top Values
action	string	3.28%	all, EXPAND, VIEW
anonymous_id	string	28.86%	c240747c-18f9-49f9-ba3c-cefec2a...
api_info.build.branch	string	100%	develop, hotfix/189-with-upsolver-c...
api_info.build.revision	string	100%	293, 380, 4290401484
api_info.build.tag	string	100%	integ-e2e, sqlake, prod-global-api
api_info.environmenttype	string	100%	prod, integ-e2e, integ
api_info.trackingversion	string	100%	2
app	string	11.39%	dbux, Sqlake
cluster_id	string	<0.01%	4be0753f-aeb0-4e8c-9a64-772d02...
cluster_name	string	<0.01%	Default Compute, testcompute2, te...
completed_item_text	string	0.52%	(, ;, JOB
completed_item_type	string	0.52%	Unit, Constant, Field
completion_options[]	string	0.81%	NOT, (, -
connection_id	string	0.69%	6f7458ed-e476-4858-87c8-ccf0ca0...

### Value Distribution

Value	Array	First Seen	Last Seen
(	false	2022-08-24 07:12	2023-03-22 03:46
;	false	2022-08-24 07:12	2023-03-22 03:46
JOB	false	2022-08-24 07:12	2023-03-22 03:46
TABLE	false	2022-08-24 07:12	2023-03-22 03:46
)	false	2022-08-24 07:12	2023-03-22 03:46
CONNECTION	false	2022-08-24 07:12	2023-03-22 03:46
SELECT	false	2022-08-24 07:12	2023-03-22 03:46
DELETE_DATA	false	2022-08-24 07:12	2023-03-22 03:46
DROP	false	2023-01-23 07:09	2023-03-16 09:12
FROM	false	2023-01-23 06:57	2023-03-16 07:26
CREATE	false	2023-01-05 08:35	2023-03-22 03:41
MAP_COLUMNS_BY_NAME	true	2023-01-05 08:35	2023-03-22 03:45
	false	2022-08-24 07:21	2023-03-22 03:41

Not what I expected

Newly added outcome

No longer updated

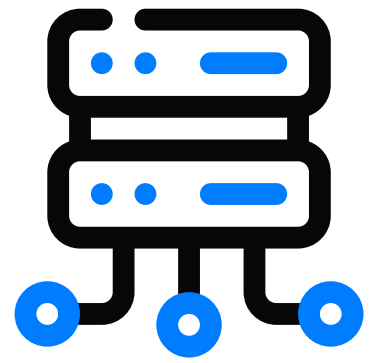
## RESOLVE AND PREVENT

```
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





# Automatic data and schema healing

Sources and targets use different data types and naming conventions. Upsolver acts as a reliable translator, preventing 100s of data bugs.



## A FEW EXAMPLES OF BUGS PREVENTED



Can't find my source field in the target



Seeing 2 columns for the same source field



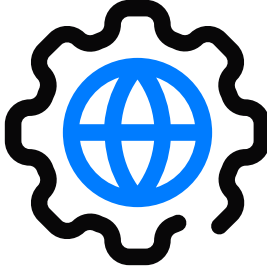
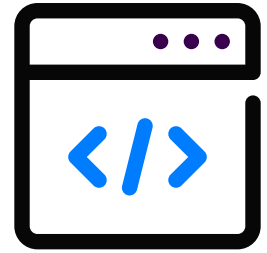

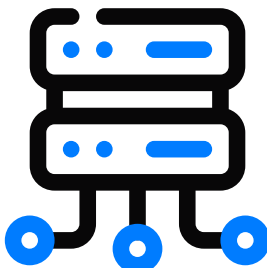


Column value is incorrect after type conversion



Replication stopped after a change in source

# Together these capabilities nullify the DIY complexity

Freeing up engineers to pursue high-value activities

	DIY - OPEN SOURCE (SPARK-BASED)	DIY - DW ADD-ON (SNOWPIPE)	SELF-SERVE (UPSOLVER)
 <p><b>Single tool</b> Single tool for E(T)L instead of many moving pieces</p>	✗	✗	✓
 <p><b>Coding optional</b> Coding is optional (no and low code experience)</p>	✗	✗	✓
 <p><b>Quality is built-in</b> Quality is built-in: discover, resolve, prevent</p>	✗	✗	✓
 <p><b>Automatic healing</b> Automatic data and schema correction</p>	✗	✗	✓
<p><b>Outcome</b></p>	<p>Months to implement Constant break-fix and ops</p>		<p>Hours to implement Enterprise-grade, near-zero ops</p>  

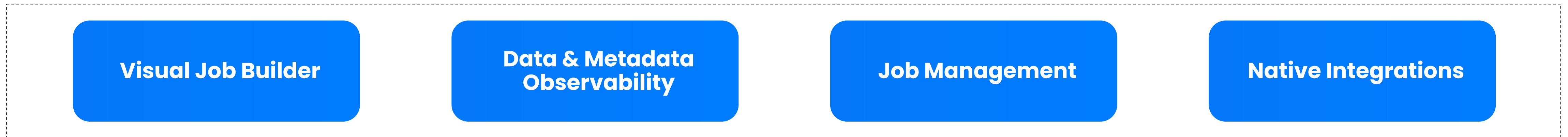


# The secret sauce powering simplicity at high-scale

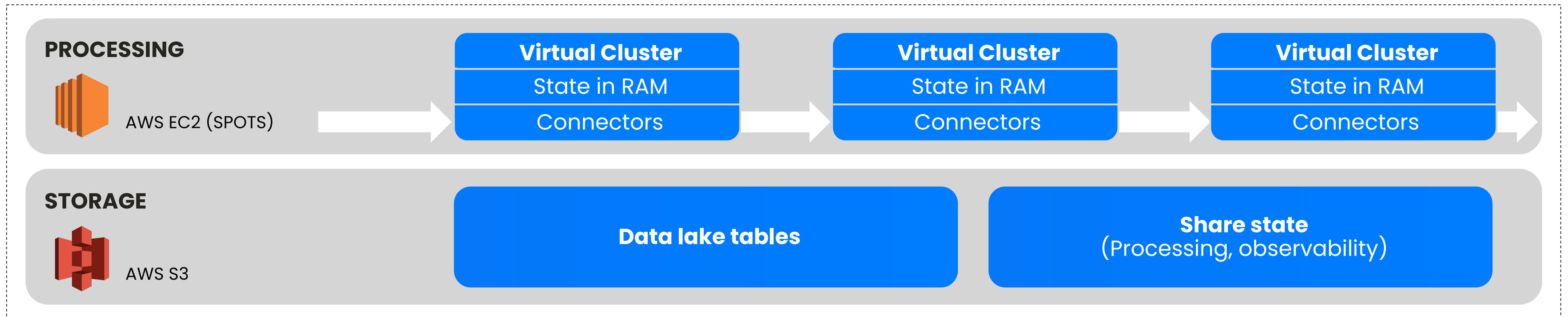


Upsolver is the only data movement solution with a **decoupled state store**, enabling a Snowflake-like cloud architecture: *shared-nothing except data on S3*

## CLOUD SERVICES



## MANAGED DATA PLANE – SAAS OR CUSTOMER VPC



# Customers choose Upsolver for simplicity at scale



APP MONETIZATION PLATFORM

**Kafka**  
to AWS ecosystem

**50 PB**  
data moved per month

**Won**  
against Spark on AWS EMR

**\$1.4 M**  
ACV



GROUP OF ADVERTISING COMPANIES

**Kafka**  
to Lake and Snowflake

**160 TB**  
data moved per month

**Won**  
against AWS Glue and Snowpipe

**\$120 K**  
ACV



CYBER SECURITY UNICORN

**CDC**  
many Postgres DBs to Snowflake

**60 TB**  
data moved per month

**Won**  
against Fivetran and DIY

**\$200 K**  
ACV



HEALTHCARE PROVIDER, FORTUNE 26

**Kafka**  
to Snowflake

**4 TB**  
data moved per month

**Won**  
against AWS Glue and Snowpipe

**\$180 K**  
ACV

**//** We want to minimize the time our engineering teams, including DevOps, spend on infrastructure and maximize the time spent developing features. **Upsolver has saved thousands of engineering hours and significantly reduced total cost of ownership**"



Seva Feldman, Vice President of Research and Development



MANUFACTURING, GLOBAL 2000

**IoT**  
to AWS

**52 TB**  
data moved per month

**Won**  
against AWS Glue

**\$225 K**  
ACV



TALENT ACQUISITION CRM

**CDC**  
Postgres DB to Snowflake

**6 TB**  
data moved per month

**Replaced**  
Fivetran

**\$73 K**  
ACV

**//** **Upsolver is like the 'easy button' for Snowflake.** We ingest data from our Kafka streams, process it for different use cases, and deliver it, all while observing how our schema and data are changing in real time."



Alexander Adam, Manager, Data Lake Cloud ETL



# Simple, risk-free evaluation

## 01 10 minute setup

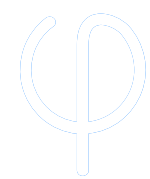
- Create an account
- Ingest using 3-step wizard
- Deploy into AWS VPC (optional)

## 02 Unlimited 14-day trial

- Any scale of data
- Any number of users
- Access to all features

## 03 Purchase

- Buy in the AWS marketplace
- Leverage existing AWS budget





$\varphi$  upsolver