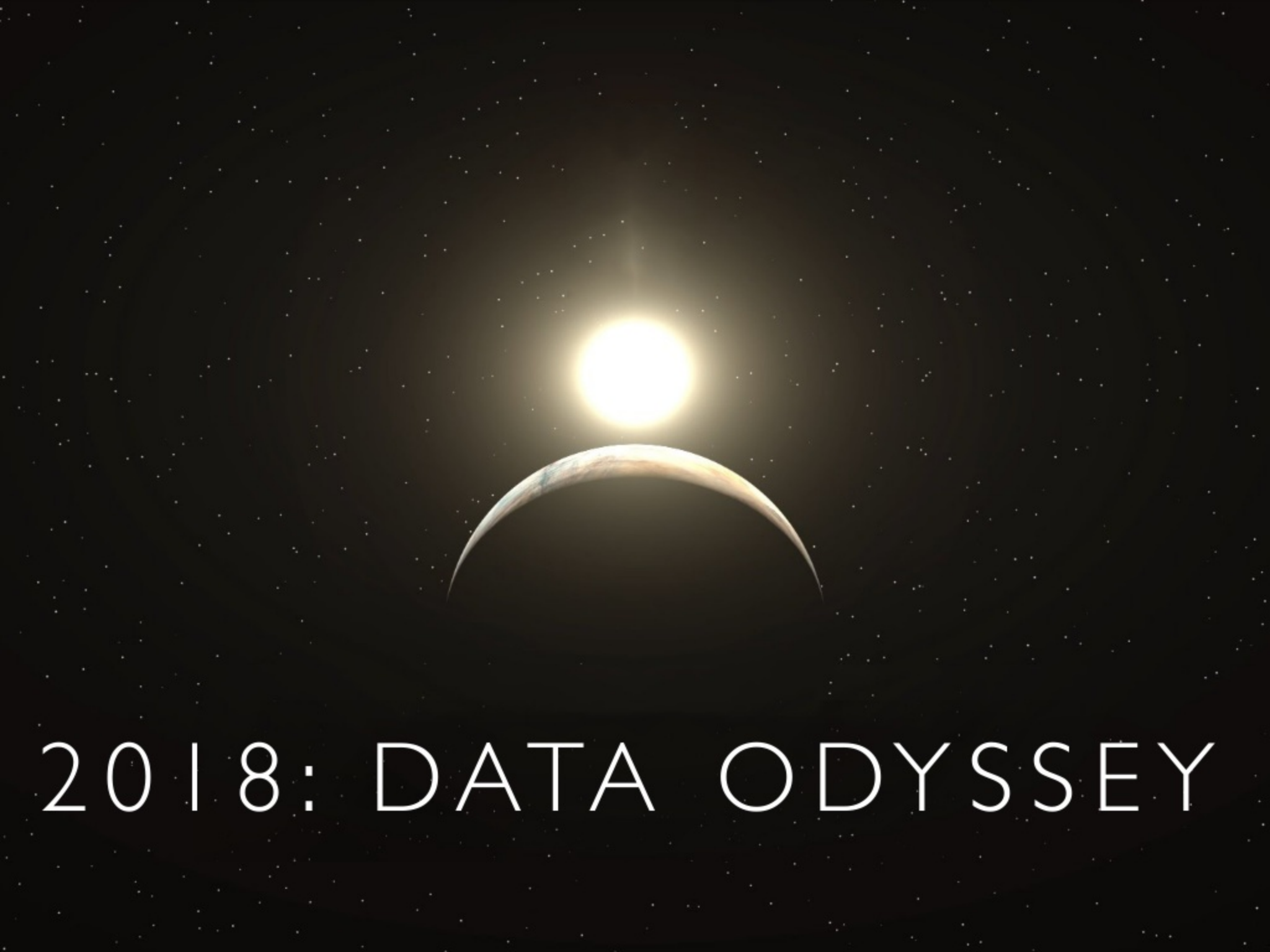
Designing a Horizontally Scalable Event-Driven Big Data Architecture with Apache Spark

Ricardo Fanjul, letgo

#SAISExp2


2018: DATA ODYSSEY

Ricardo Fanjul
Data Engineer

letgo

- Founded in 2015
- 100MM+ downloads
- 400MM+ listings

LETGO DATA PLATFORM IN NUMBERS

1 billion
Events Processed Daily

50K
Peaks of events per Second

500GB
Data daily

600+
Event Types

200TB
Storage (S3)

< 1sec
NRT Processing Time

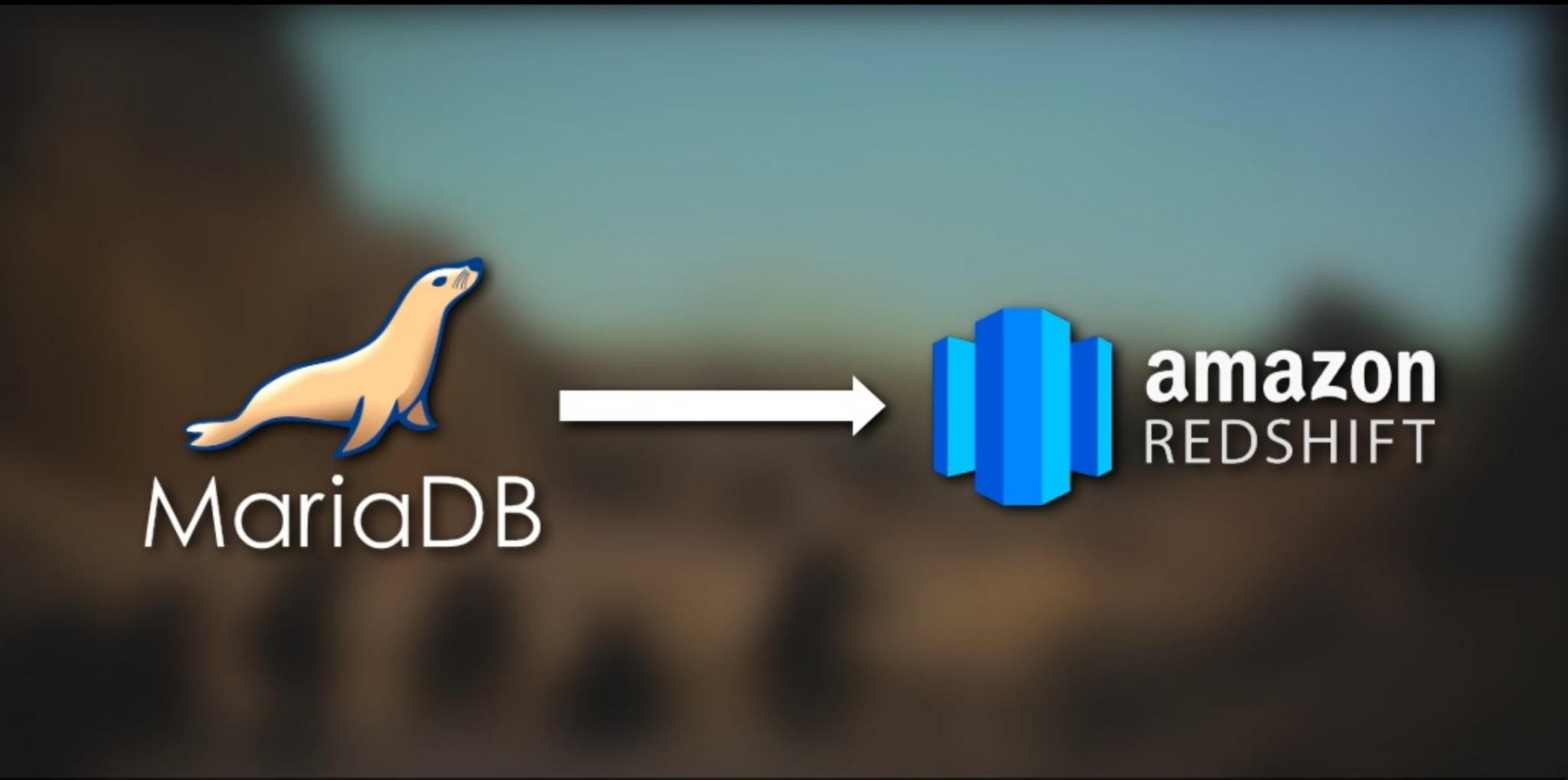

THE DAWN OF LETGO

THE DAWN OF LETGO

CLASSICAL BI PLATFORM

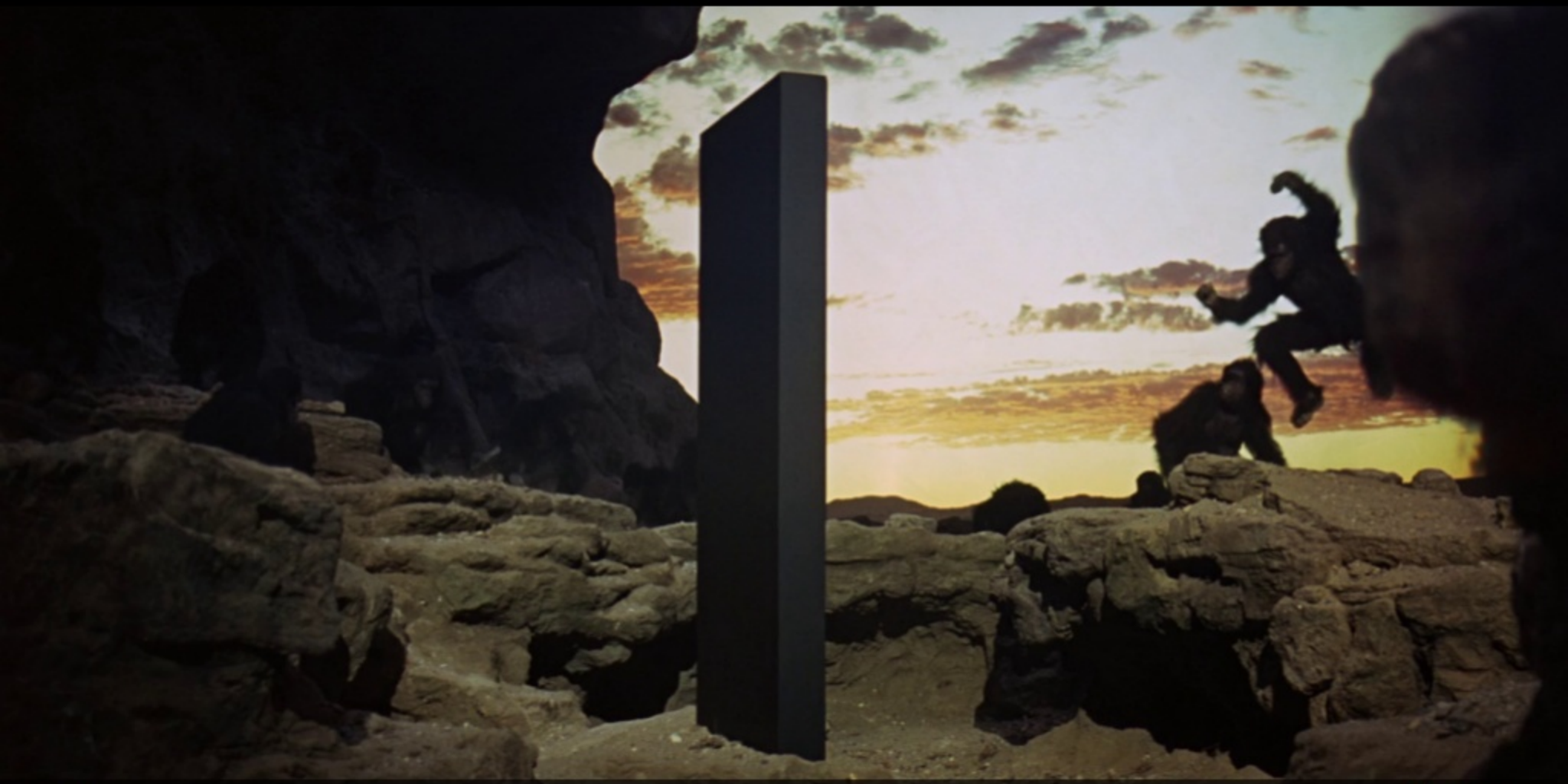
THE DAWN OF LETGO

CLASSICAL BI PLATFORM



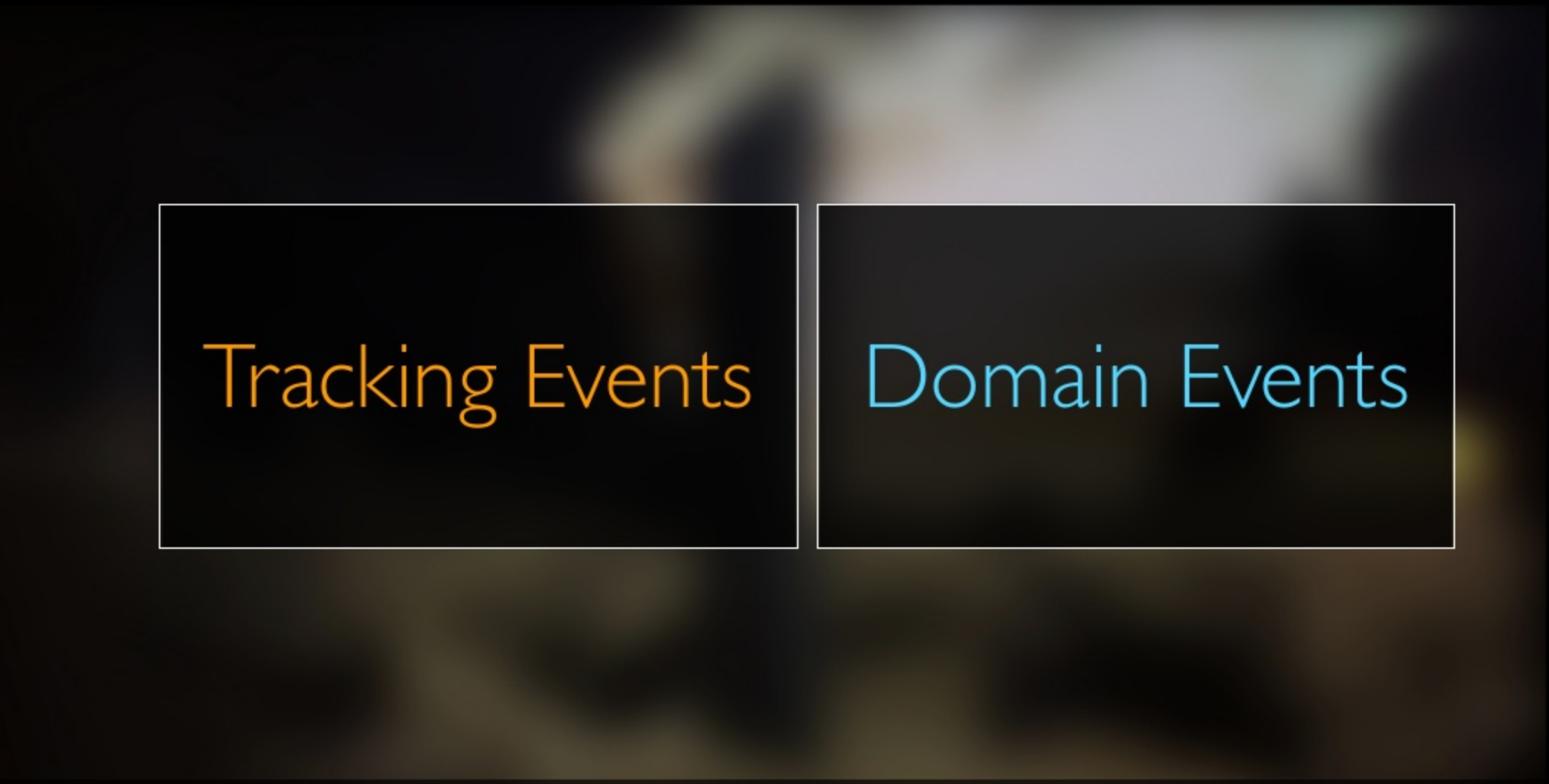
THE DAWN OF LETGO

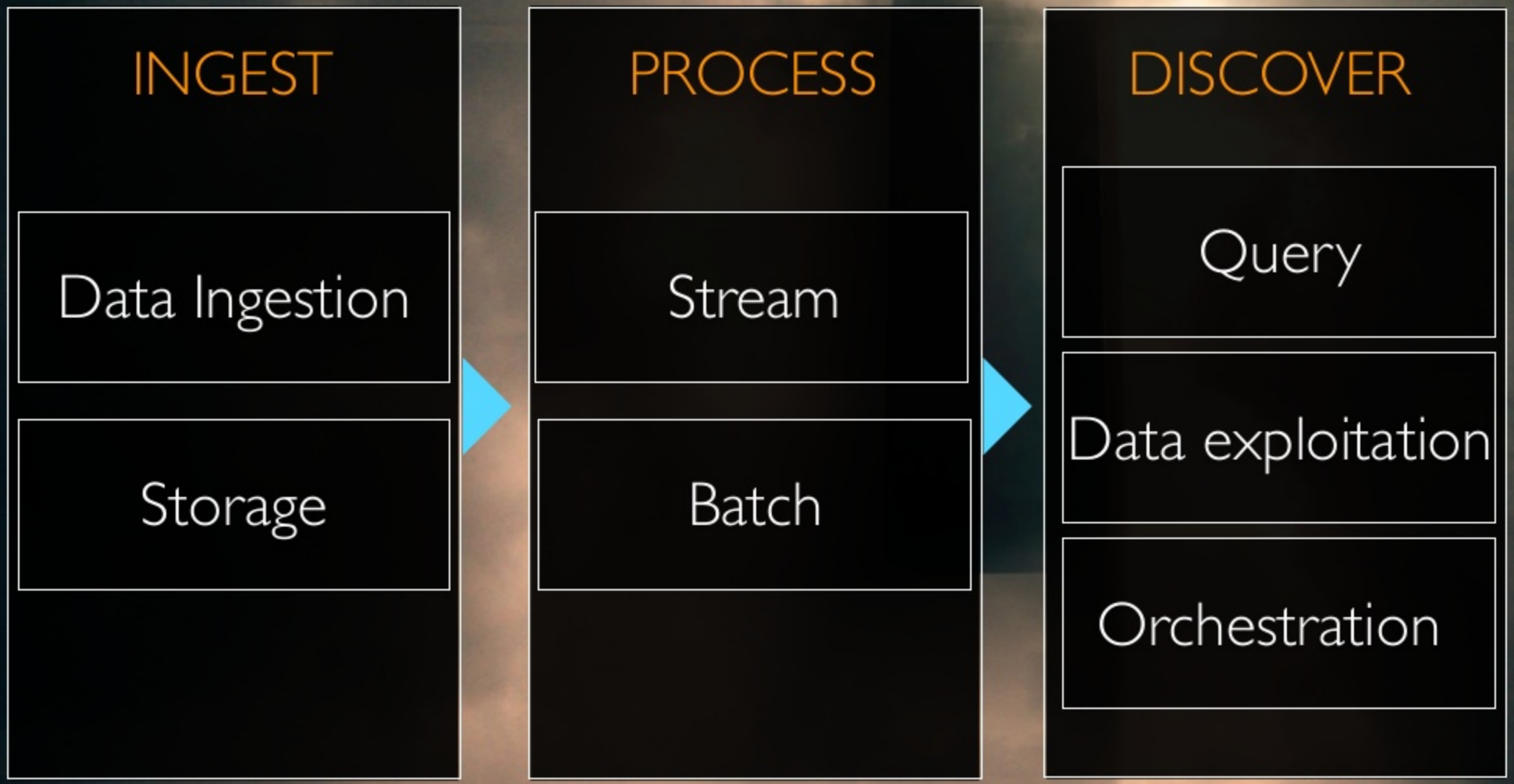
MOVING TO μ-SERVICES AND EVENTS

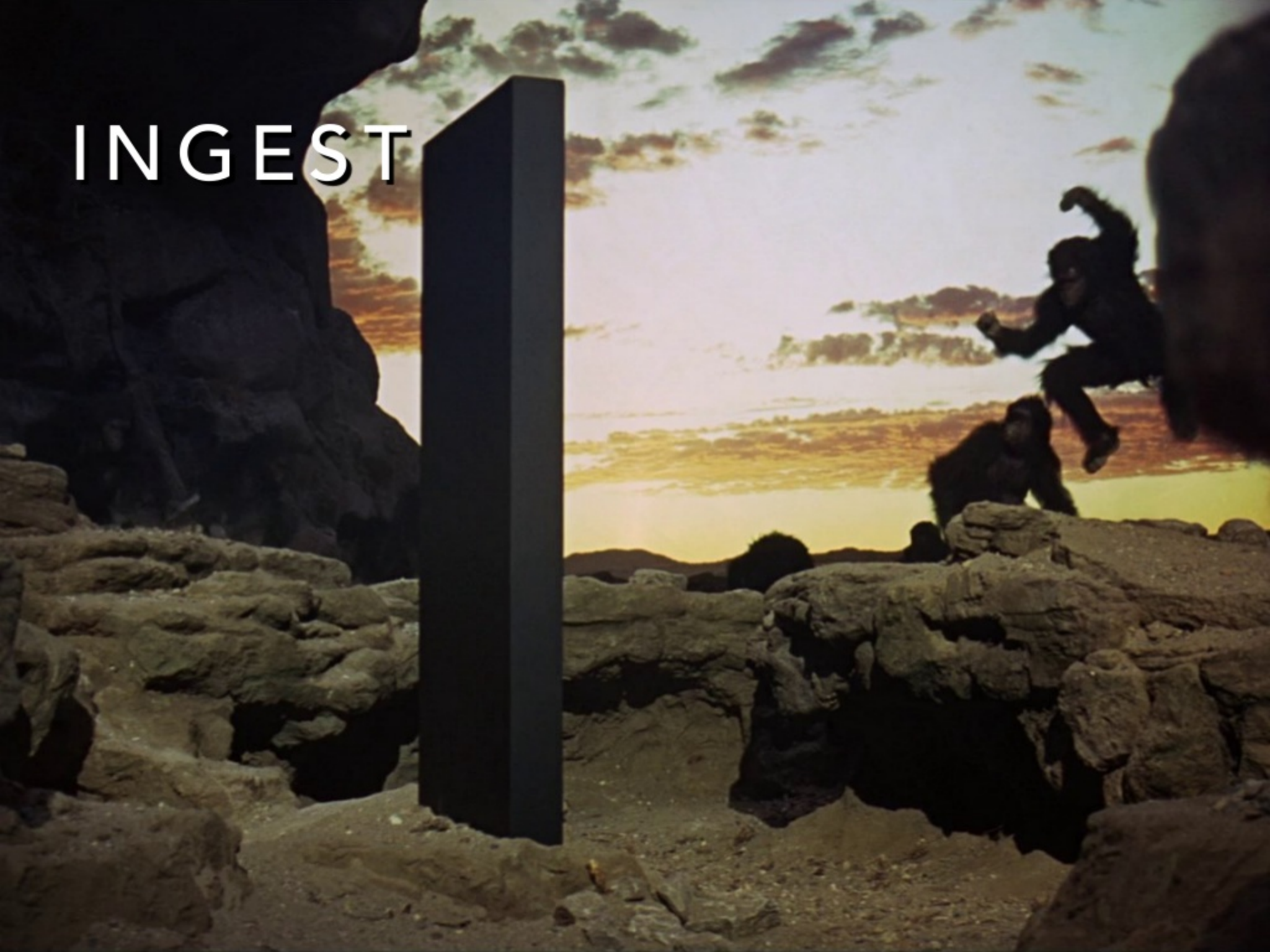


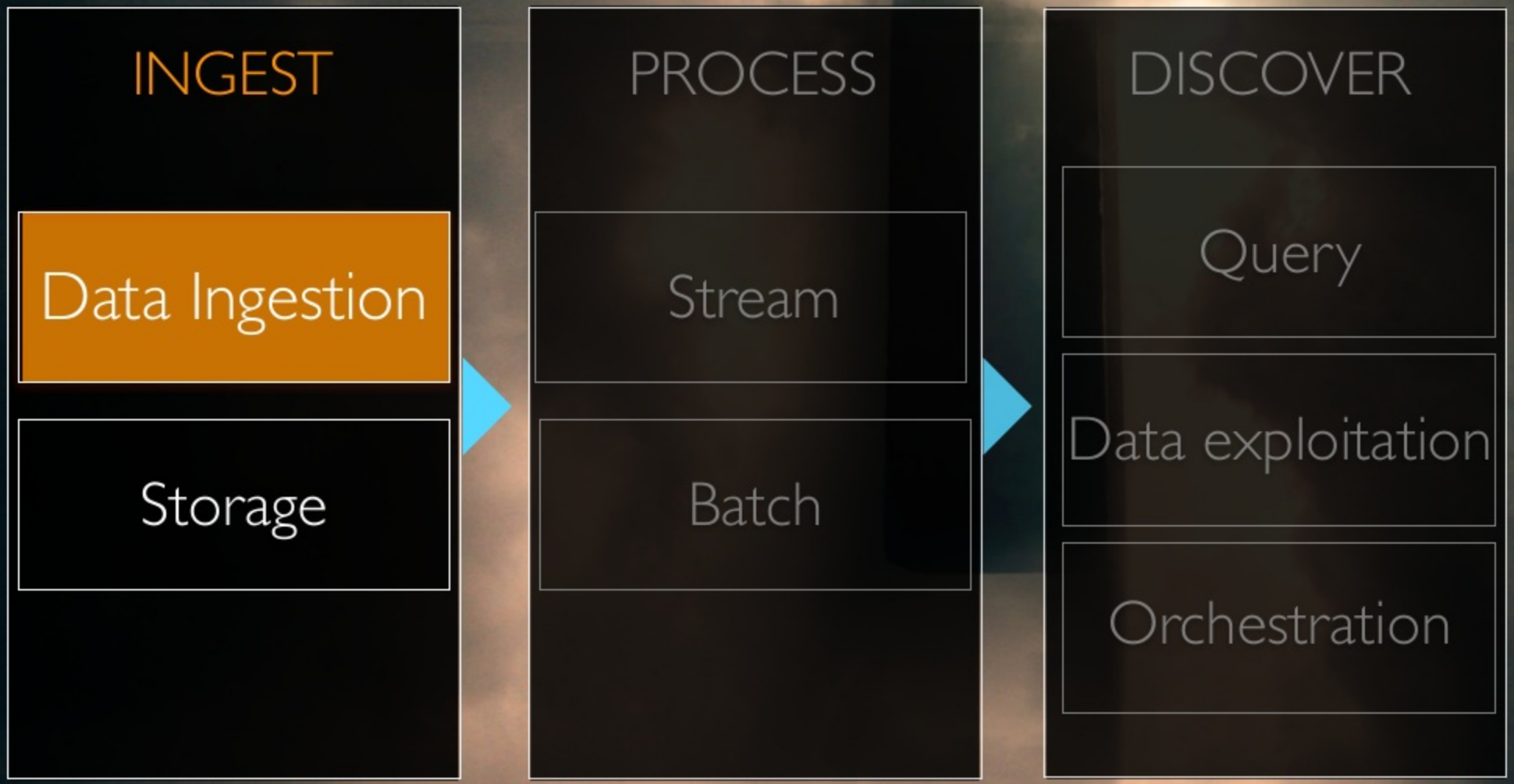
THE DAWN OF LETGO

MOVING TO μ-SERVICES AND EVENTS




INGEST
Data Ingestion
Storage
PROCESS
Stream
Batch
DISCOVER
Query
Data exploitation
Orchestration


INGEST


INGEST
Data Ingestion
Storage
PROCESS
Stream
Batch
DISCOVER
Query
Data exploitation
Orchestration

DATA INGESTION

OUR GOAL

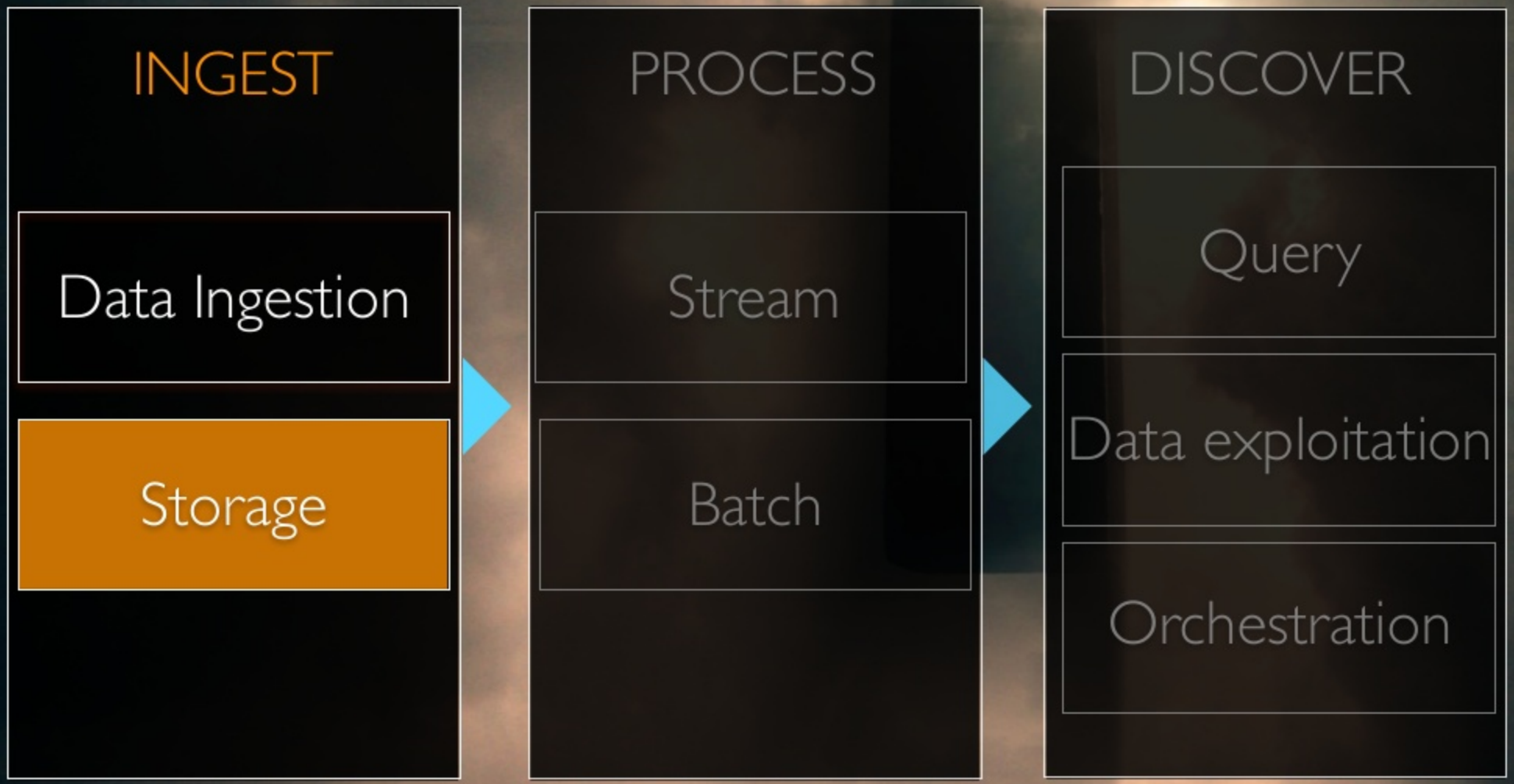
DATA INGESTION

THE DISCOVERY

DATA INGESTION

KAFKA CONNECT


THE JOURNEY BEGINS


INGEST
Data Ingestion
Storage
PROCESS
Stream
Batch
DISCOVER
Query
Data exploitation
Orchestration

STORAGE

BUILDING THE DATA LAKE

STORAGE

BUILDING THE DATA LAKE

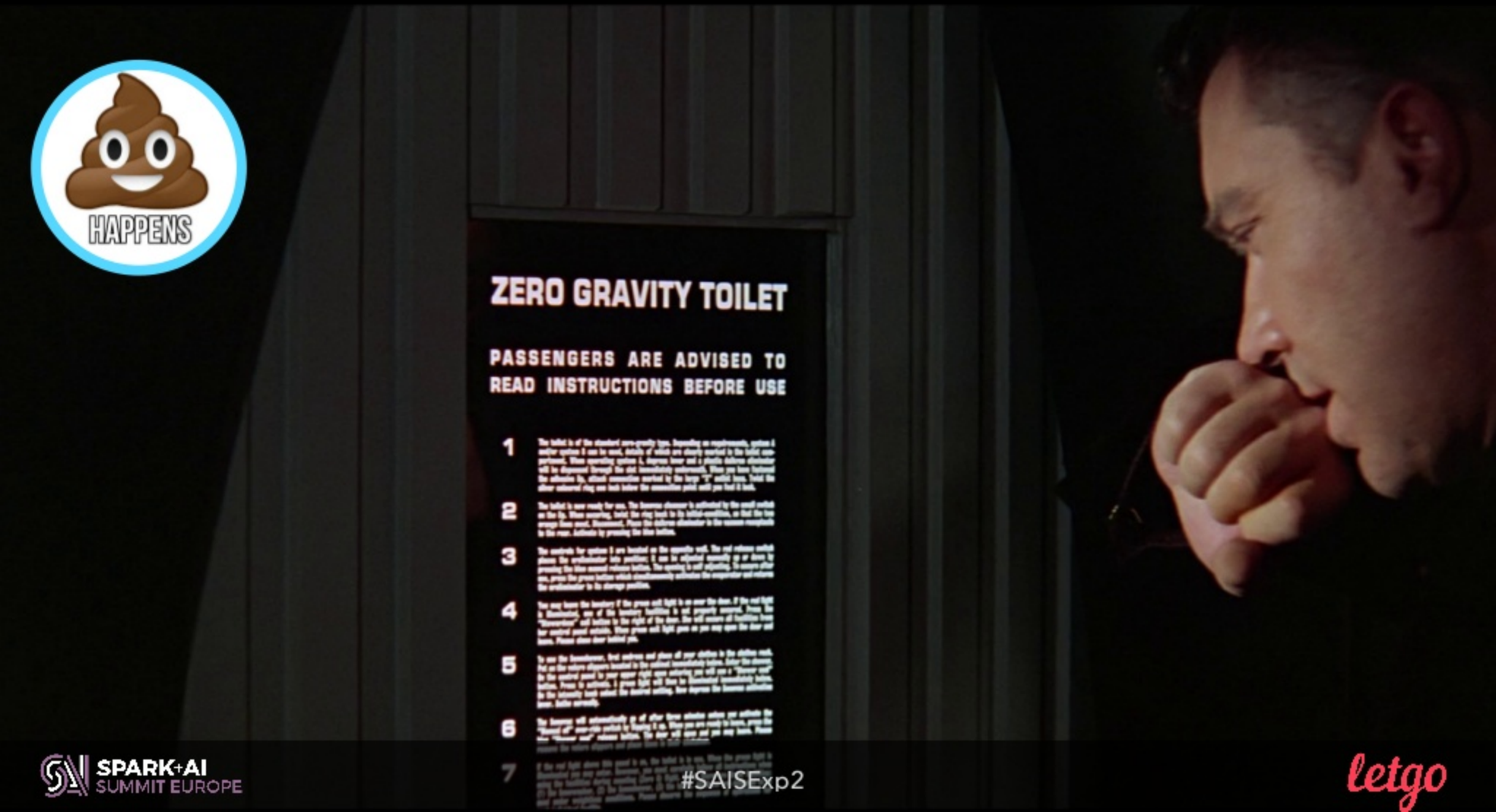
We want to store all events coming from Kafka to S3.

STORAGE

BUILDING THE DATA LAKE

STORAGE

SOMETIMES... SHIT HAPPENS

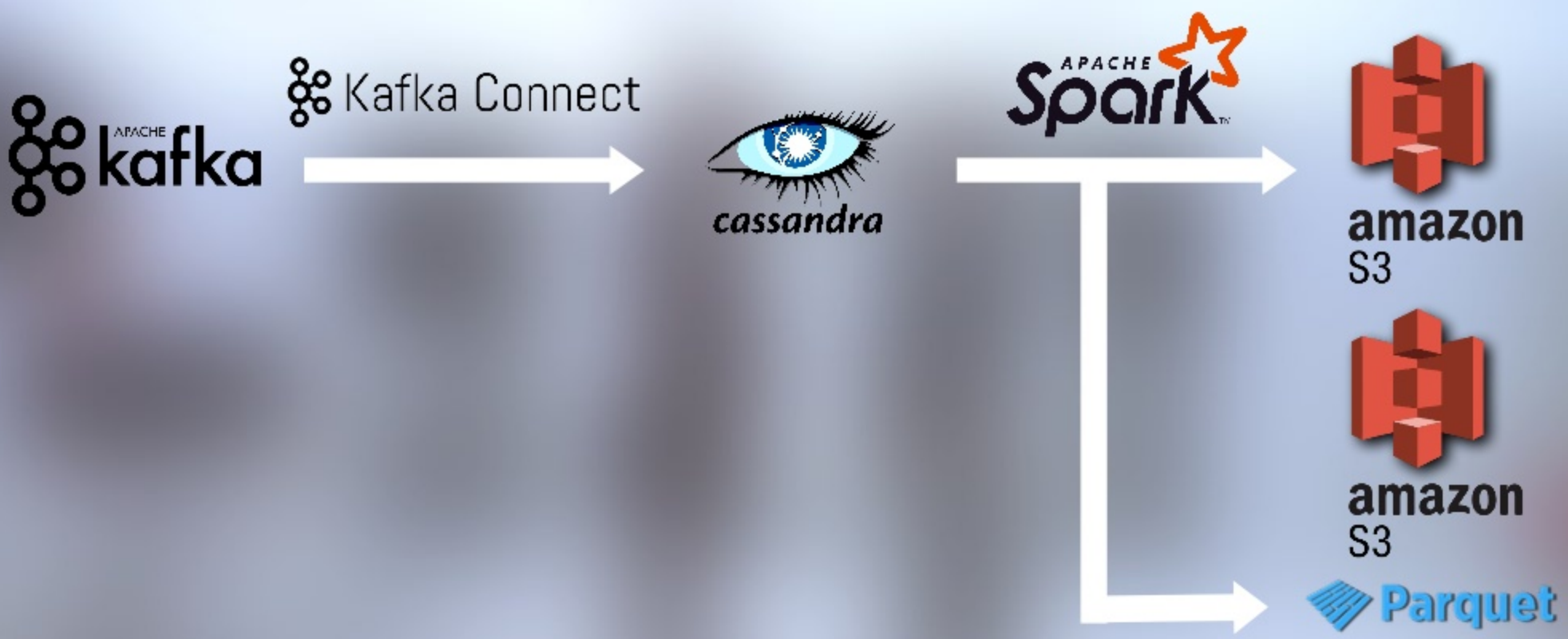


STORAGE

DUPLICATED EVENTS

STORAGE

DUPLICATED EVENTS

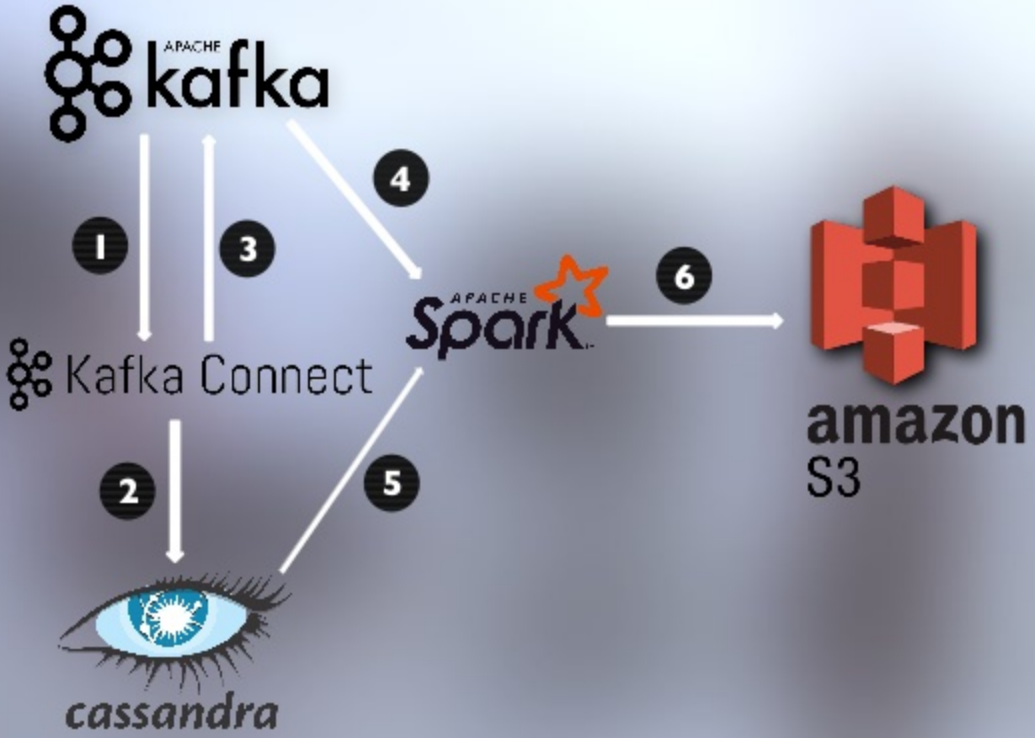


STORAGE

(VERY) LATE EVENTS

STORAGE

(VERY) LATE EVENTS



1. Read batch of events from Kafka.
2. Write each event to Cassandra.
3. Write "dirty hours" to compact topic: Key=(event_type, hour).
4. Read dirty "hours" topic.
5. Read all events with dirty hours.
6. Store in S3

Dirty Buckets

STORAGE

S3 PROBLEMS

STORAGE

S3 PROBLEMS

VS

amazon
S3

SOME S3 BIG DATA PROBLEMS:

1. Eventual consistency
2. Very slow renames

STORAGE

S3 PROBLEMS: EVENTUAL CONSISTENCY

STORAGE

S3 PROBLEMS: EVENTUAL CONSISTENCY

S3GUARD

FileSystem Operations

S3AFileSystem

The Hadoop FileSystem for Amazon S3

DynamoDB Client

S3 Client

2 WRITE fs metadata

1 READ fs metadata

1 WRITE object data

2 LIST object info

3 LIST object data

Write Path 1 2

Read Path 1 2 3

amazon DynamoDB

amazon S3

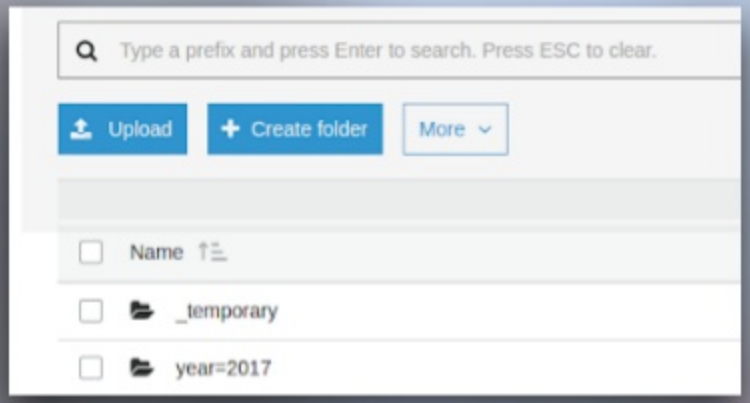
STORAGE

S3 PROBLEMS: SLOW RENAMES

Summary

	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks
Active(2)	4	107.1 KB / 23.1 GB	0.0 B	8	0	0	8	8
Dead(10)	4	128.3 KB / 95.5 GB	0.0 B	80	0	0	1436	1436
Total(12)	8	235.4 KB / 118.6 GB	0.0 B	88	0	0	1444	1444

Executors



¿Job freeze?

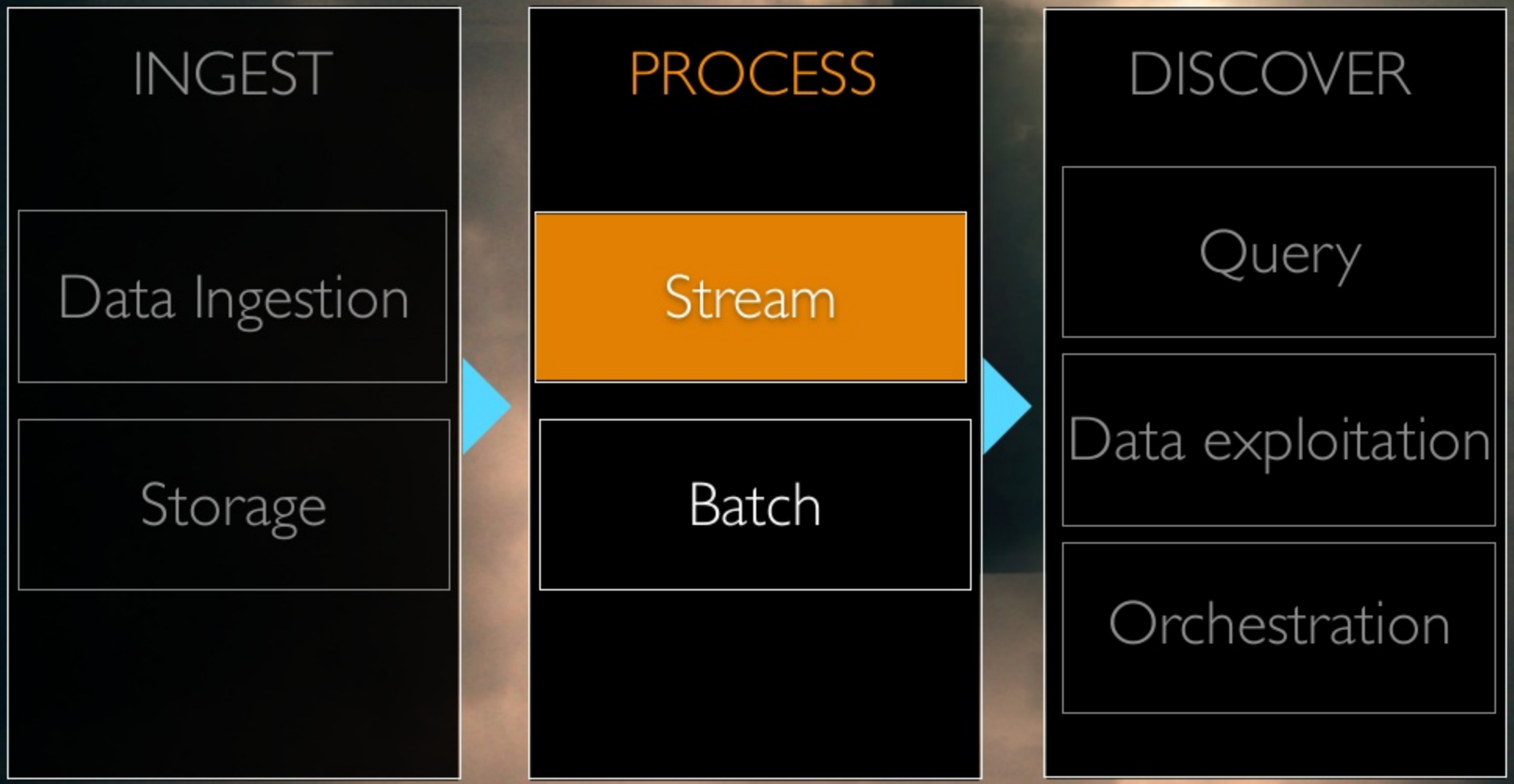
STORAGE

S3 PROBLEMS: SLOW RENAMES

New **Hadoop 3.1** S3A committers:

- Directory
- Partitioned
- Magic


PROCESS

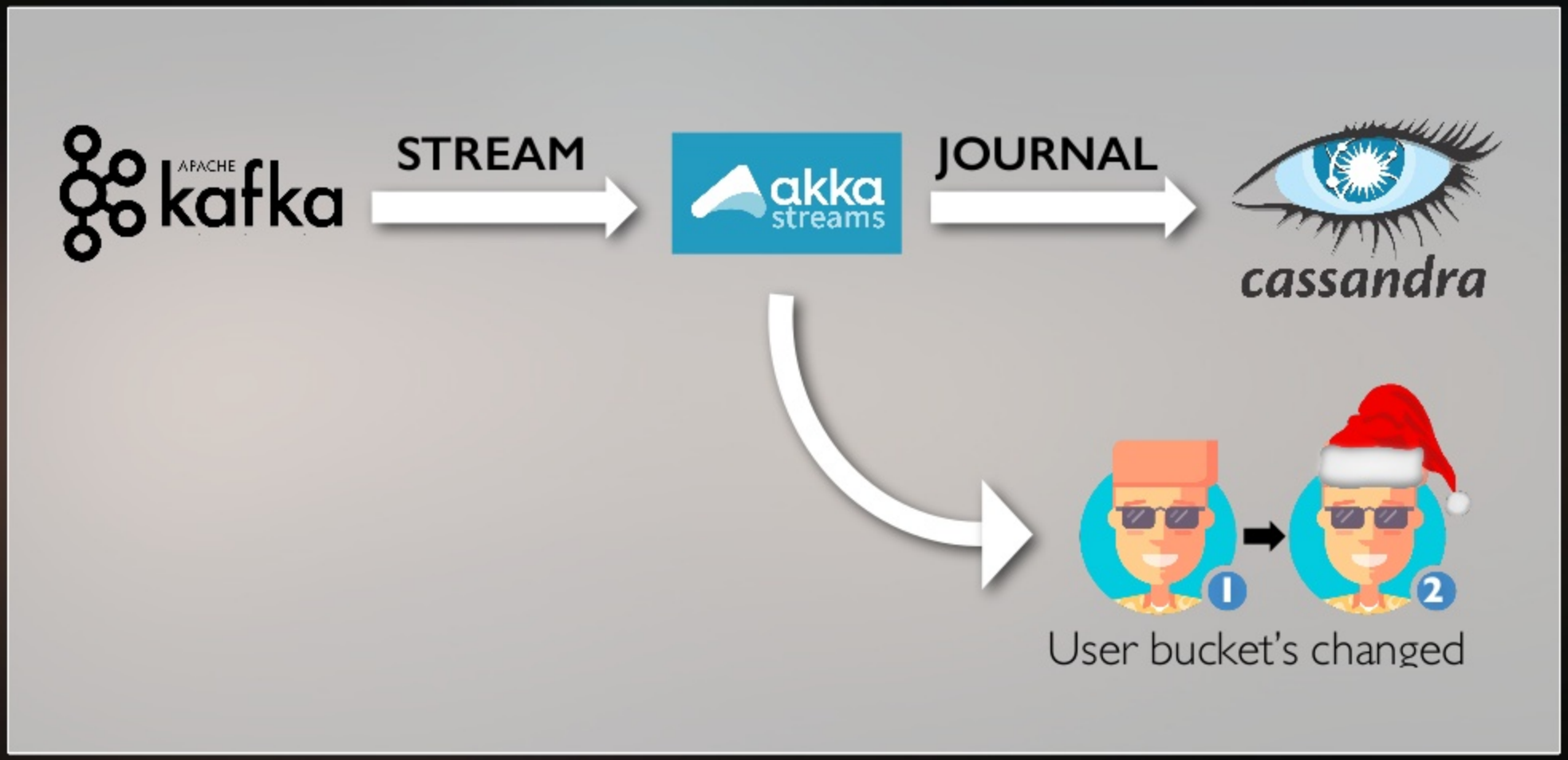

INGEST
Data Ingestion
Storage
PROCESS
Stream
Batch
DISCOVER
Query
Data exploitation
Orchestration

STREAM

REAL TIME USER SEGMENTATION

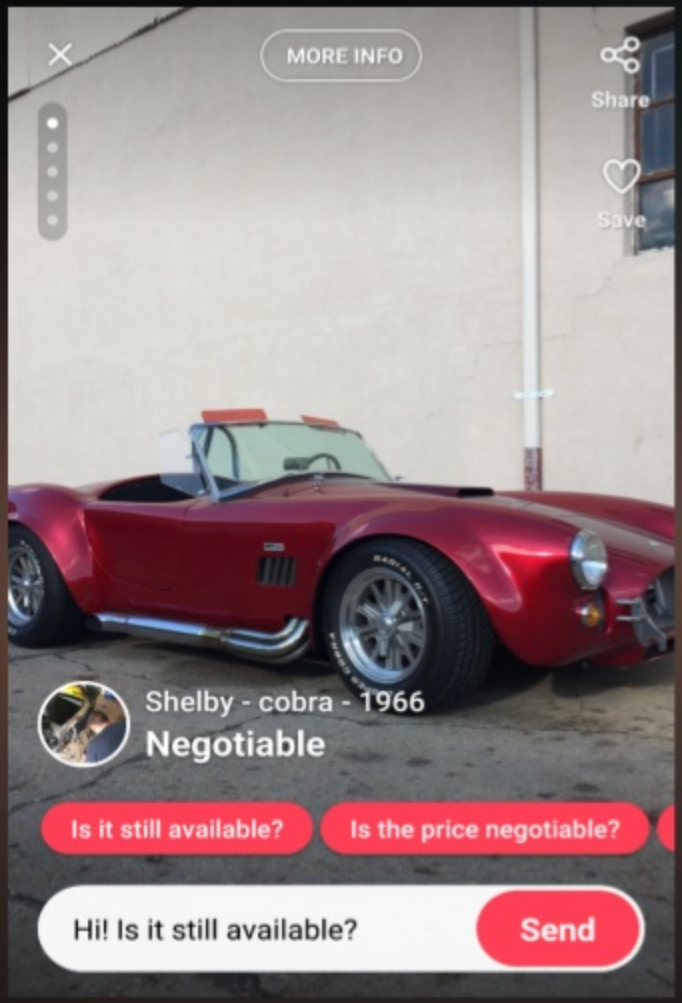
STREAM

REAL TIME USER SEGMENTATION



STREAM

REAL TIME PATTERN DETECTION



Is it still available?

Is the price negotiable?

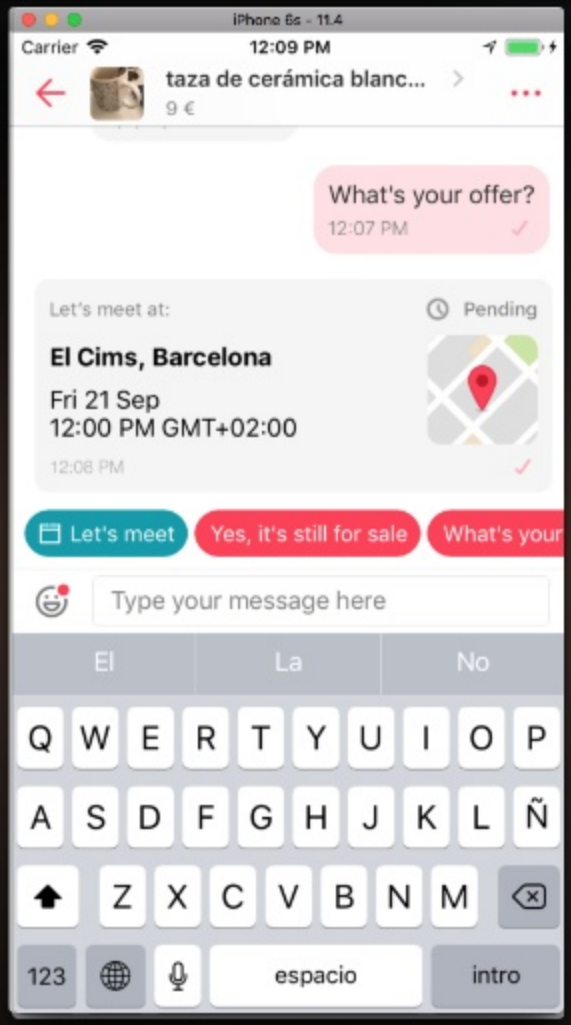
What condition is it in?

I offer you….$

Could we meet at……?

STREAM

REAL TIME PATTERN DETECTION

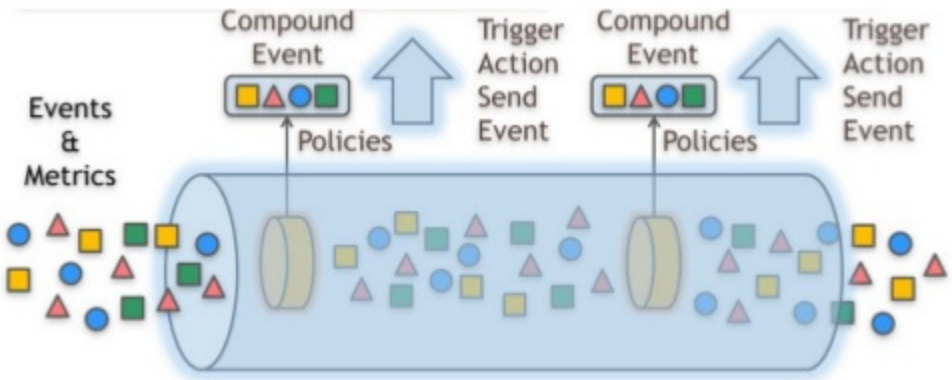


Structured data

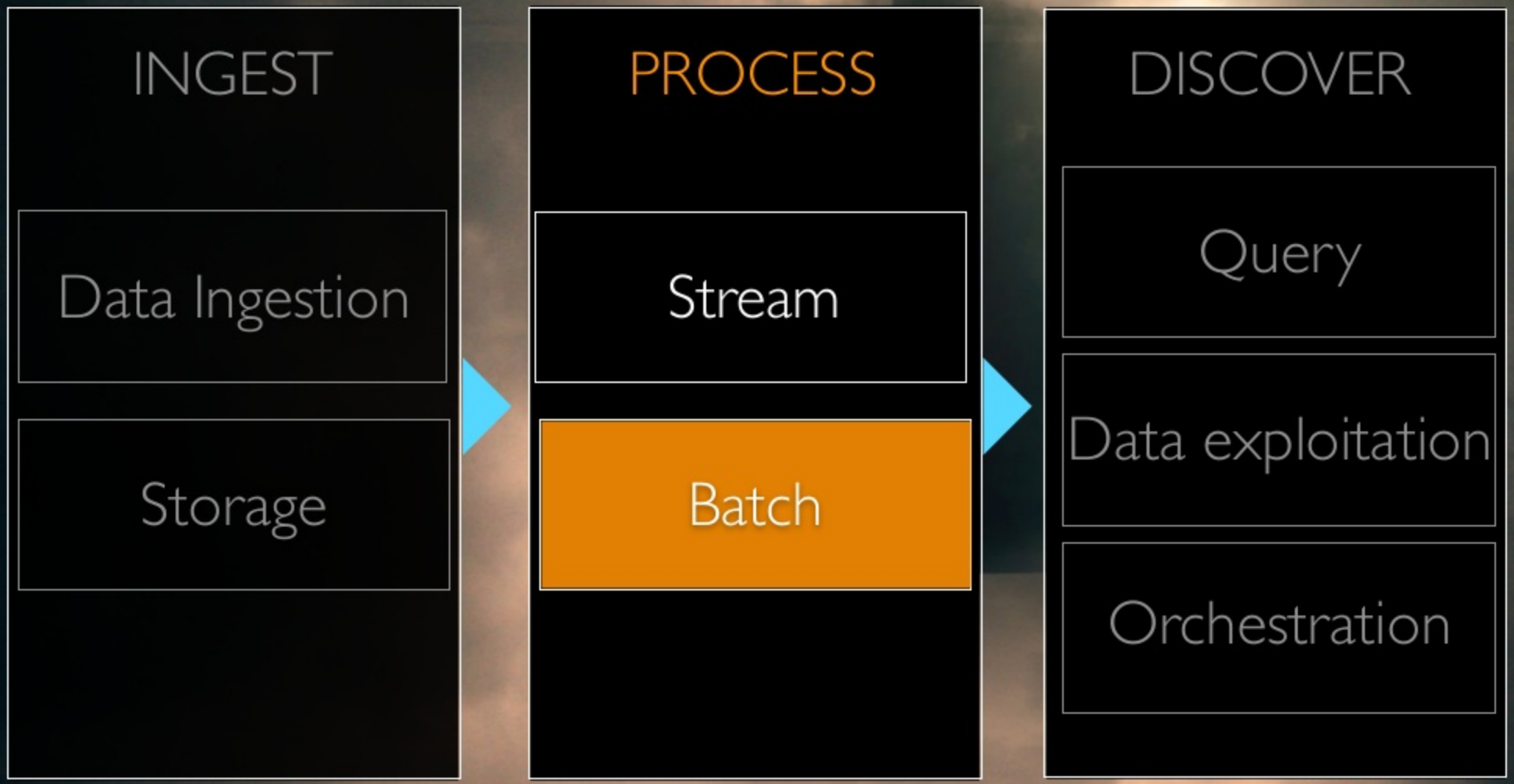
```
{
  "type": "meeting_proposal",
  "properties": {
    "location_name": “Letgo HQ",
    "geo": {
      "lat": "41.390205",
      "lon": "2.154007"
    },
    "date": "1511193820350",
    "meeting_id": "23213213213"
  }
}
```

STREAM

REAL TIME PATTERN DETECTION



Meeting proposed + meeting accepted = emit accepted-meeting event
Meeting proposed + nothing in X time = "You have a proposal to meet"

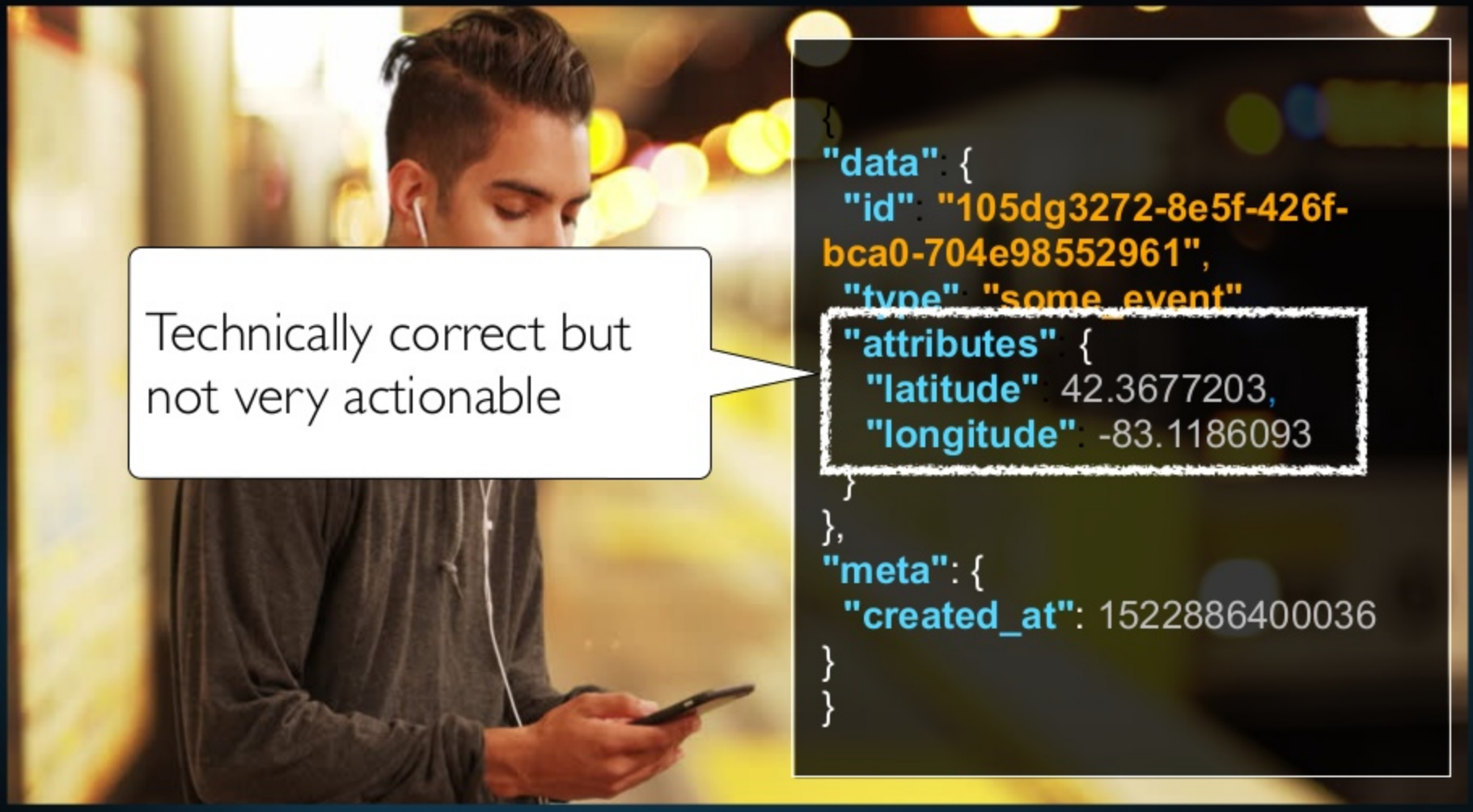

INGEST
Data Ingestion
Storage
PROCESS
Stream
Batch
DISCOVER
Query
Data exploitation
Orchestration

BATCH

GEODATA ENRICHMENT

BATCH

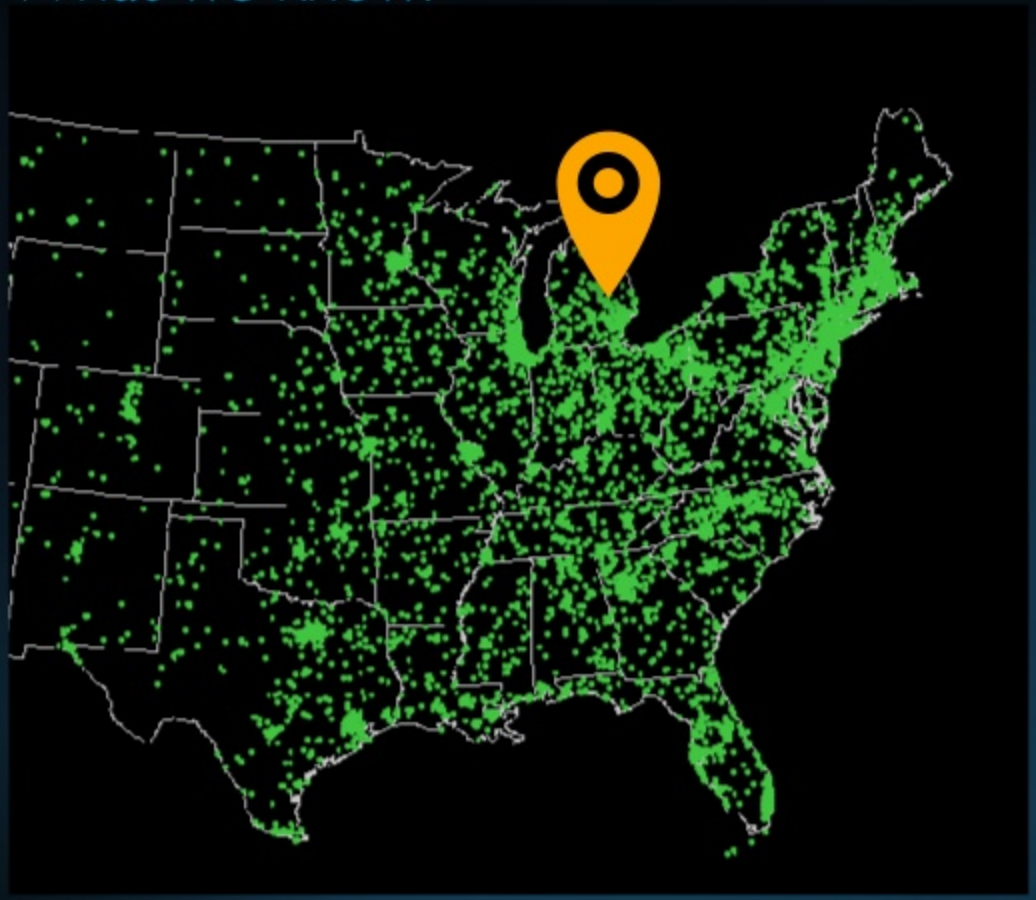
GEODATA ENRICHMENT



BATCH

GEODATA ENRICHMENT

What we know:



(42.3677203, -83.1186093)

- City: Detroit
- Postal code: 48206
- State: Michigan
- DMA: Detroit
- Country: US

BATCH

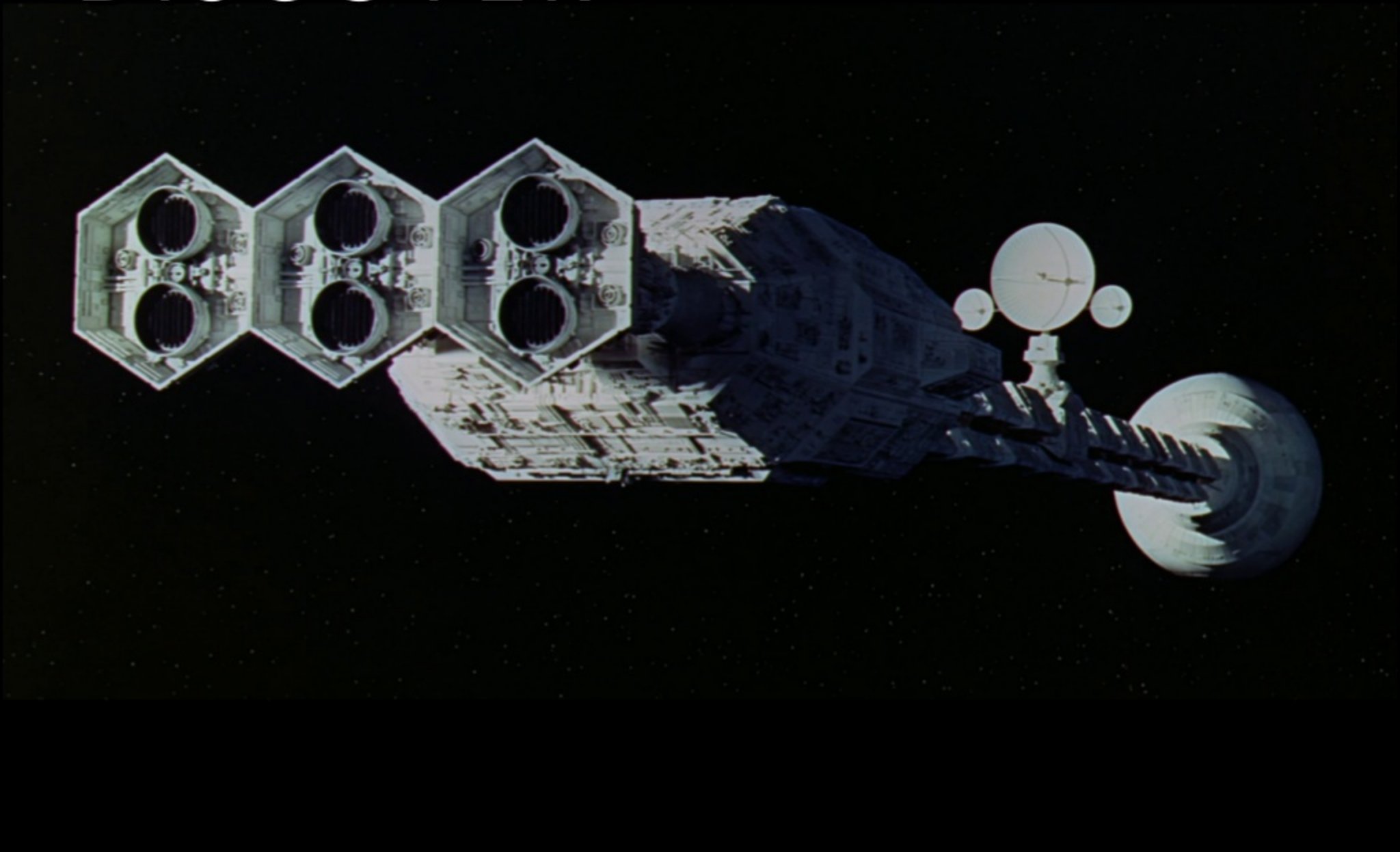
GEODATA ENRICHMENT

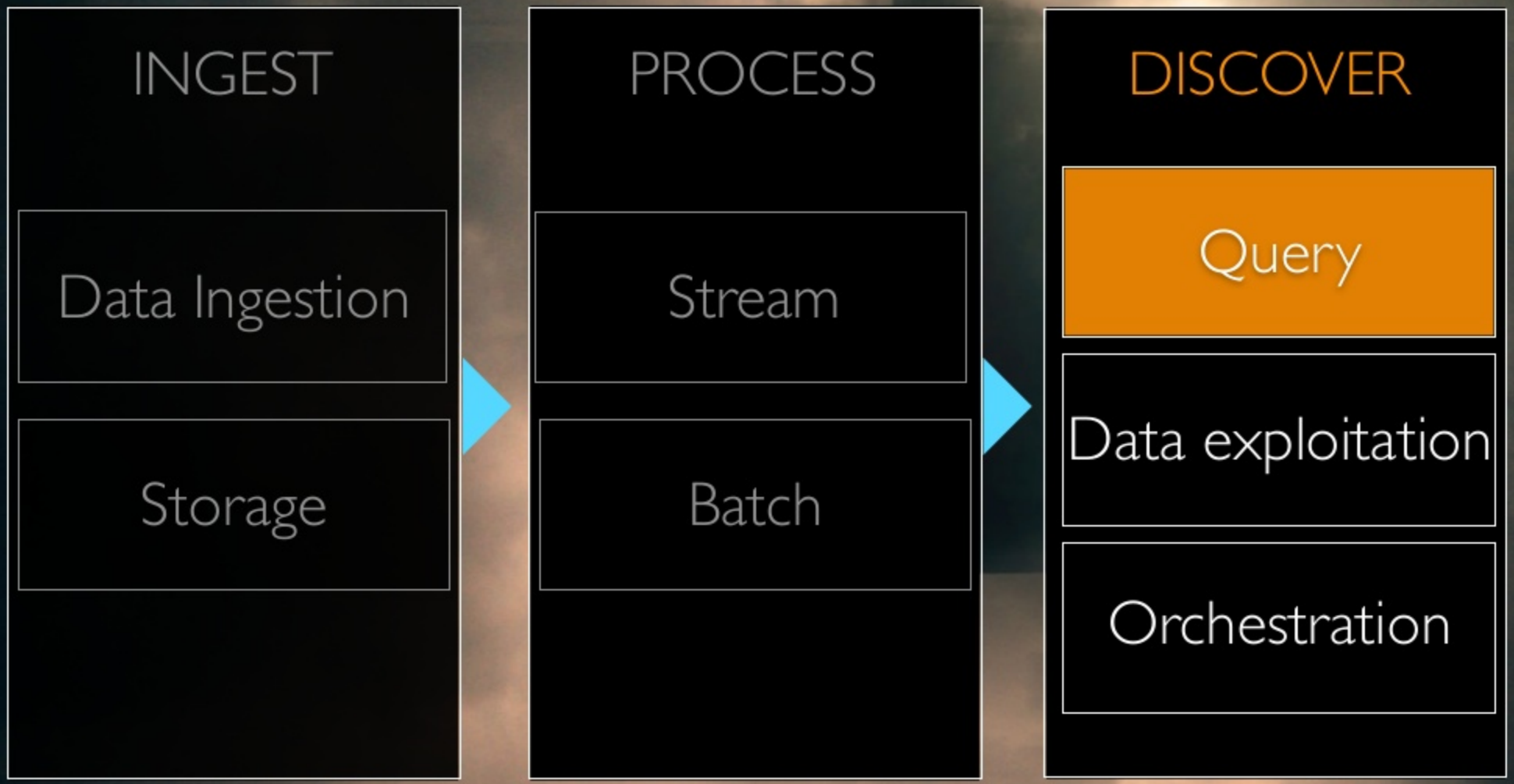
How we do it:

- Populating JTS indices from WKT polygon data
- Custom Spark SQL UDF


```
SELECT geodata.dma_name
    geodata.dma_number AS dma_number
    geodata.city       AS city
    geodata.state      AS state
    geodata.zip_code   AS zip_code
FROM   (
      SELECT
        geodata(longitude, latitude) AS geodata
      FROM   ....
    )
```

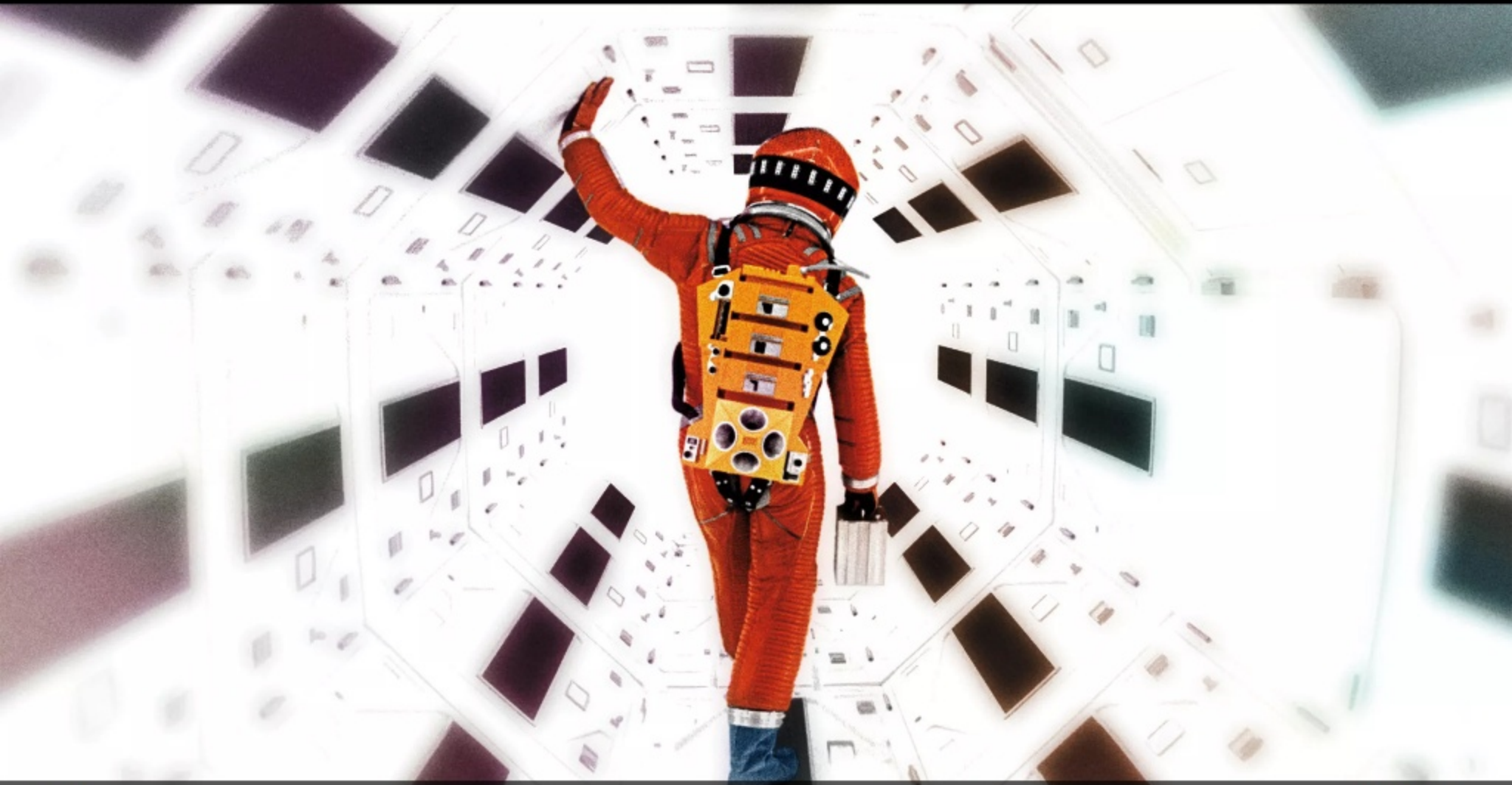
DISCOVER




INGEST
Data Ingestion
Storage
PROCESS
Stream
Batch
DISCOVER
Query
Data exploitation
Orchestration

QUERY

QUERYING DATA



QUERY

QUERYING DATA

QUERY

QUERYING DATA

Amazon Aurora

QUERY

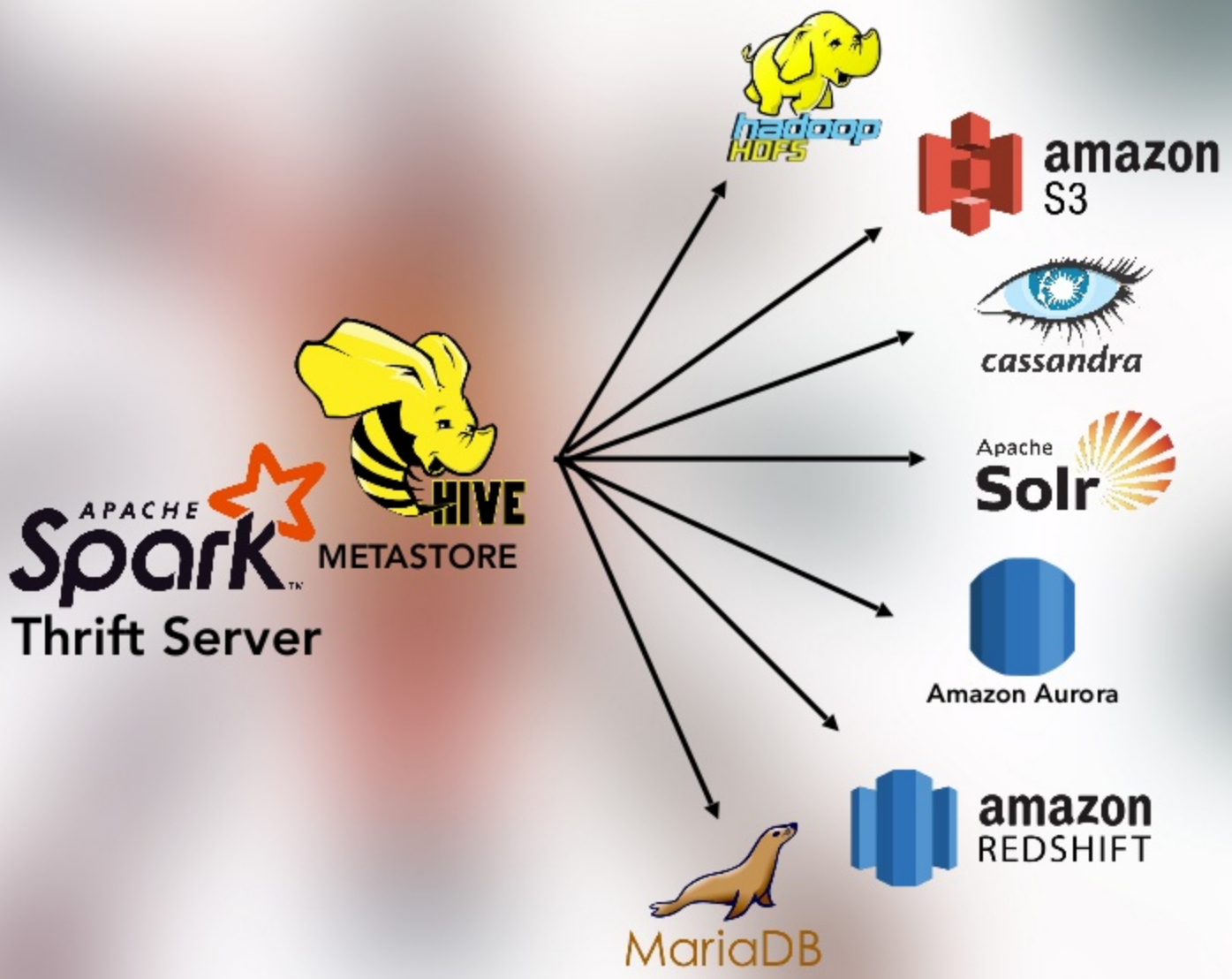
QUERYING DATA

QUERY

QUERYING DATA

QUERY

QUERYING DATA



QUERY

QUERYING DATA


```
CREATE TABLE IF NOT EXISTS
database_name.table_name(
some_column STRING,
...
dt DATE
)
USING json
PARTITIONED BY (`dt`)
```


```
CREATE EXTERNAL TABLE IF NOT EXISTS
database_name.table_name(
some_column STRING...,
dt DATE
)
PARTITIONED BY (`dt`)
USING PARQUET
LOCATION 's3a://bucket-name/database_name/table_name'
```


```
CREATE TEMPORARY VIEW table_name
USING org.apache.spark.sql.cassandra
OPTIONS (
table "table_name",
keyspace "keyspace_name")
```


```
CREATE TABLE IF NOT EXISTS database_name.table_name
using com.databricks.spark.redshift
options (
dbtable 'schema.redshift_table_name',
tempdir 's3a://redshift-temp/',
url 'jdbc:redshift://xxxx.redshift.amazonaws.com:5439/letgo?
user=xxx&password=xxx',
forward_spark_s3_credentials 'true')
```

QUERY

QUERYING DATA

CREATE TABLE …
STORED AS…

70%
Higher performance!

CREATE TABLE …
USING [parquet,json,csv…]

QUERY

QUERYING DATA: BATCHES WITH SQL

1

Creating the table

2

Inserting data

QUERY

QUERYING DATA: BATCHES WITH SQL

1

Creating the table

```
CREATE EXTERNAL TABLE IF NOT EXISTS database.some_name(
  user_id STRING,
  column_b STRING,
...
)
USING PARQUET
PARTITIONED BY (`dt` STRING)
LOCATION 's3a://example/some_table'
```

QUERY

QUERYING DATA: BATCHES WITH SQL

2

Inserting data

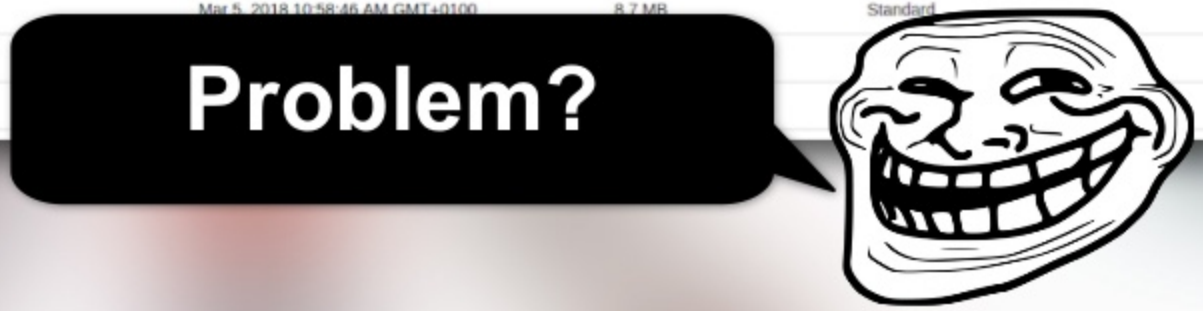
```
INSERT OVERWRITE TABLE database.some_name PARTITION(dt)
SELECT
user_id,
column_b,
dt
FROM other_table
...
```

QUERY

QUERYING DATA: BATCHES WITH SQL

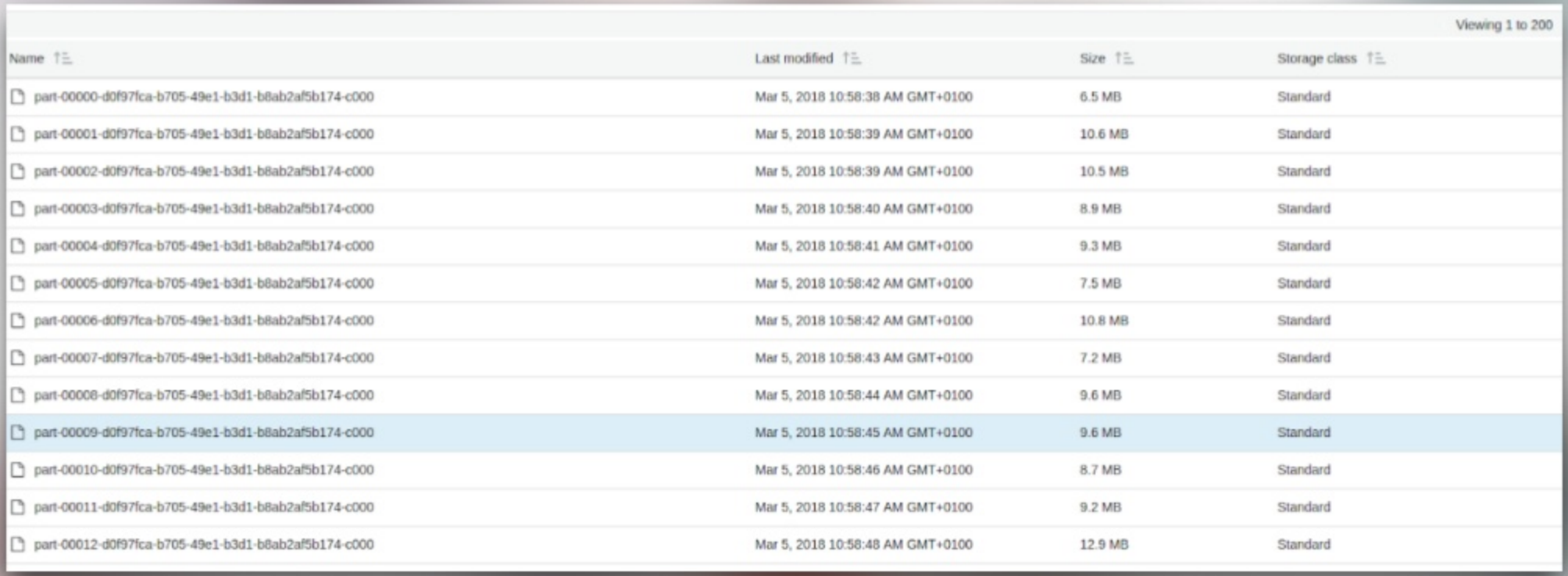
Viewing 1 to 200

Name	Last modified	Size	Storage class
part-00000-d0f97fca-b705-49e1-b3d1-b8ab2af5b174-c000	Mar 5, 2018 10:58:38 AM GMT+0100	6.5 MB	Standard
part-00001-d0f97fca-b705-49e1-b3d1-b8ab2af5b174-c000	Mar 5, 2018 10:58:39 AM GMT+0100	10.6 MB	Standard
part-00002-d0f97fca-b705-49e1-b3d1-b8ab2af5b174-c000	Mar 5, 2018 10:58:39 AM GMT+0100	10.5 MB	Standard
part-00003-d0f97fca-b705-49e1-b3d1-b8ab2af5b174-c000	Mar 5, 2018 10:58:40 AM GMT+0100	8.9 MB	Standard
part-00004-d0f97fca-b705-49e1-b3d1-b8ab2af5b174-c000	Mar 5, 2018 10:58:41 AM GMT+0100	9.3 MB	Standard
part-00005-d0f97fca-b705-49e1-b3d1-b8ab2af5b174-c000	Mar 5, 2018 10:58:42 AM GMT+0100	7.5 MB	Standard
part-00006-d0f97fca-b705-49e1-b3d1-b8ab2af5b174-c000	Mar 5, 2018 10:58:42 AM GMT+0100	10.8 MB	Standard
part-00007-d0f97fca-b705-49e1-b3d1-b8ab2af5b174-c000	Mar 5, 2018 10:58:43 AM GMT+0100	7.2 MB	Standard
part-00008-d0f97fca-b705-49e1-b3d1-b8ab2af5b174-c000	Mar 5, 2018 10:58:44 AM GMT+0100	9.6 MB	Standard
part-00009-d0f97fca-b705-49e1-b3d1-b8ab2af5b174-c000	Mar 5, 2018 10:58:45 AM GMT+0100	9.6 MB	Standard
part-00010-d0f97fca-b705-49e1-b3d1-b8ab2af5b174-c000	Mar 5, 2018 10:58:46 AM GMT+0100	8.7 MB	Standard
part-00011-d0f97fca-b705-49e1-b3d1-b8ab2af5b174-c000			
part-00012-d0f97fca-b705-49e1-b3d1-b8ab2af5b174-c000			



QUERY

QUERYING DATA: BATCHES WITH SQL



Viewing 1 to 200

Name	Last modified	Size	Storage class
part-00000-d0f97fca-b705-49e1-b3d1-b8ab2af5b174-c000	Mar 5, 2018 10:58:38 AM GMT+0100	6.5 MB	Standard
part-00001-d0f97fca-b705-49e1-b3d1-b8ab2af5b174-c000	Mar 5, 2018 10:58:39 AM GMT+0100	10.6 MB	Standard
part-00002-d0f97fca-b705-49e1-b3d1-b8ab2af5b174-c000	Mar 5, 2018 10:58:39 AM GMT+0100	10.5 MB	Standard
part-00003-d0f97fca-b705-49e1-b3d1-b8ab2af5b174-c000	Mar 5, 2018 10:58:40 AM GMT+0100	8.9 MB	Standard
part-00004-d0f97fca-b705-49e1-b3d1-b8ab2af5b174-c000	Mar 5, 2018 10:58:41 AM GMT+0100	9.3 MB	Standard
part-00005-d0f97fca-b705-49e1-b3d1-b8ab2af5b174-c000	Mar 5, 2018 10:58:42 AM GMT+0100	7.5 MB	Standard
part-00006-d0f97fca-b705-49e1-b3d1-b8ab2af5b174-c000	Mar 5, 2018 10:58:42 AM GMT+0100	10.8 MB	Standard
part-00007-d0f97fca-b705-49e1-b3d1-b8ab2af5b174-c000	Mar 5, 2018 10:58:43 AM GMT+0100	7.2 MB	Standard
part-00008-d0f97fca-b705-49e1-b3d1-b8ab2af5b174-c000	Mar 5, 2018 10:58:44 AM GMT+0100	9.6 MB	Standard
part-00009-d0f97fca-b705-49e1-b3d1-b8ab2af5b174-c000	Mar 5, 2018 10:58:45 AM GMT+0100	9.6 MB	Standard
part-00010-d0f97fca-b705-49e1-b3d1-b8ab2af5b174-c000	Mar 5, 2018 10:58:46 AM GMT+0100	8.7 MB	Standard
part-00011-d0f97fca-b705-49e1-b3d1-b8ab2af5b174-c000	Mar 5, 2018 10:58:47 AM GMT+0100	9.2 MB	Standard
part-00012-d0f97fca-b705-49e1-b3d1-b8ab2af5b174-c000	Mar 5, 2018 10:58:48 AM GMT+0100	12.9 MB	Standard

200 files because default value of
“spark.sql.shuffle.partition”

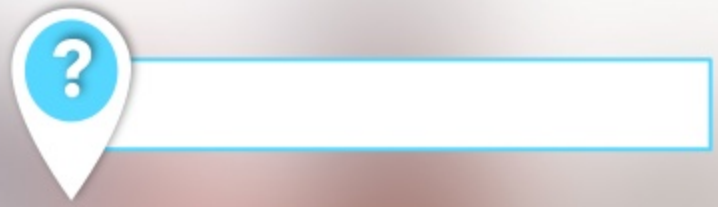
QUERY

QUERYING DATA: BATCHES WITH SQL

```
INSERT OVERWRITE TABLE database.some_name PARTITION(dt)
SELECT
user_id,
column_b,
dt
FROM other_table
...
```

QUERY

QUERYING DATA: BATCHES WITH SQL



DISTRIBUTE BY (dt):
Only one file not Sorted

CLUSTERED BY (dt, user_id, column_b):
Multiple files

DISTRIBUTE BY (dt) SORT BY (user_id, column_b):
Only one file sorted by user_id, column_b.
Good for joins using this properties.

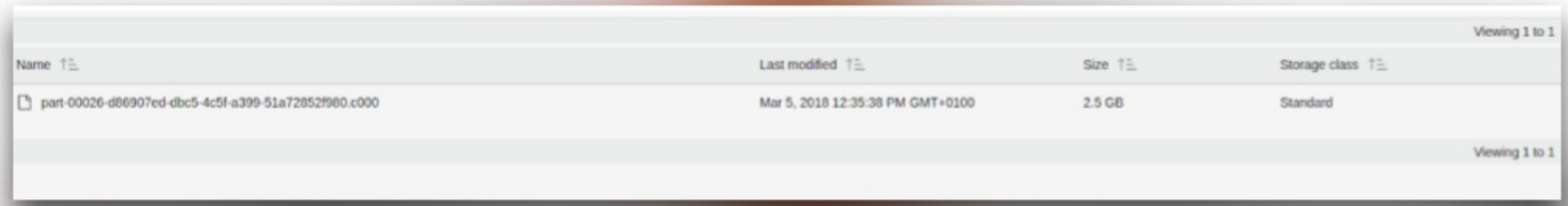
QUERY

QUERYING DATA: BATCHES WITH SQL

```
INSERT OVERWRITE TABLE database.some_name
PARTITION(dt)
SELECT
user_id,
column_b,
dt
FROM other_table
...
DISTRIBUTE BY (dt) SORT BY (user_id)
```

QUERY

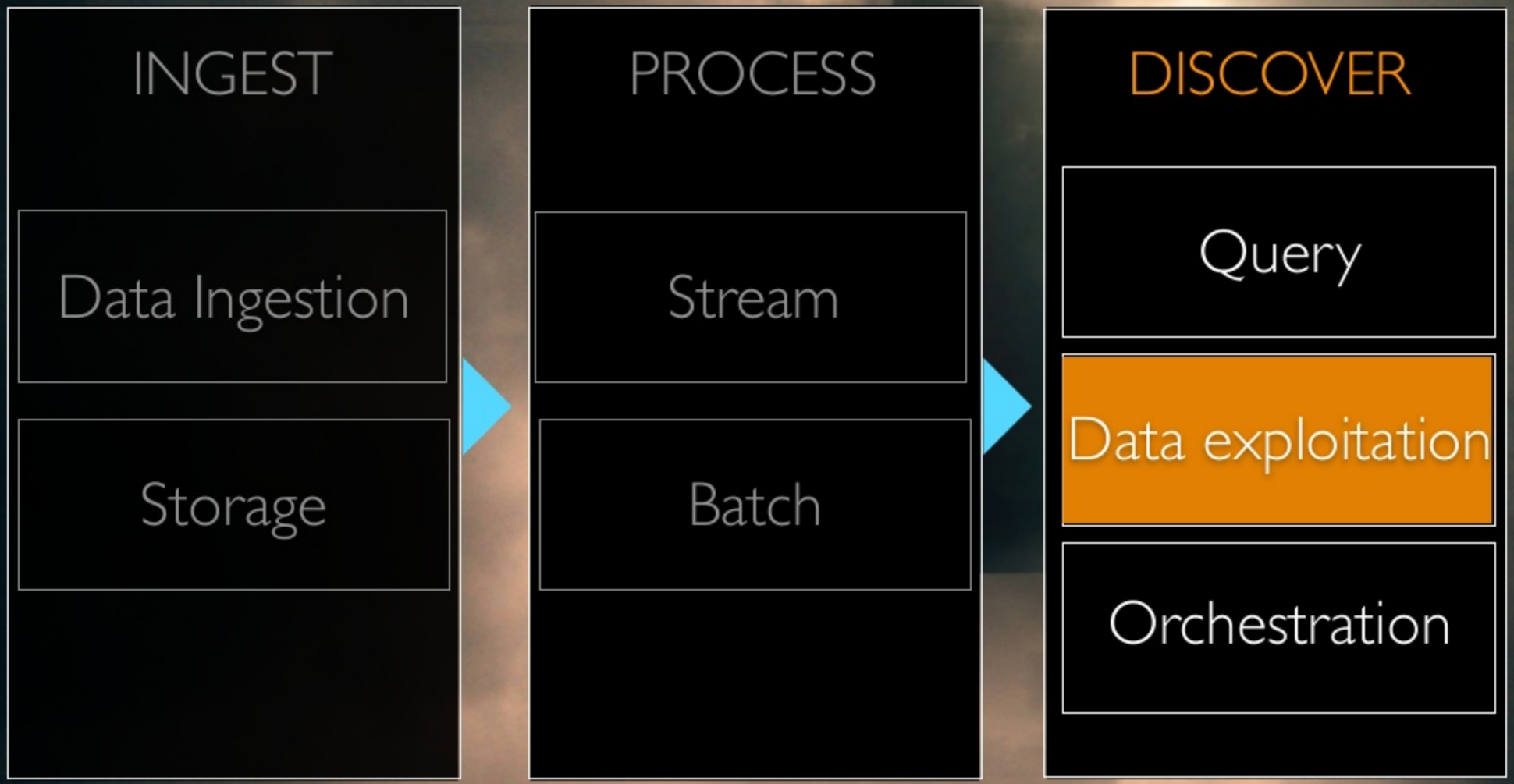
QUERYING DATA: BATCHES WITH SQL



Viewing 1 to 1

Name	Last modified	Size	Storage class
part-00026-d86907ed-dbc5-4c5f-a399-51a72852f980.c000	Mar 5, 2018 12:35:38 PM GMT+0100	2.5 GB	Standard

Viewing 1 to 1

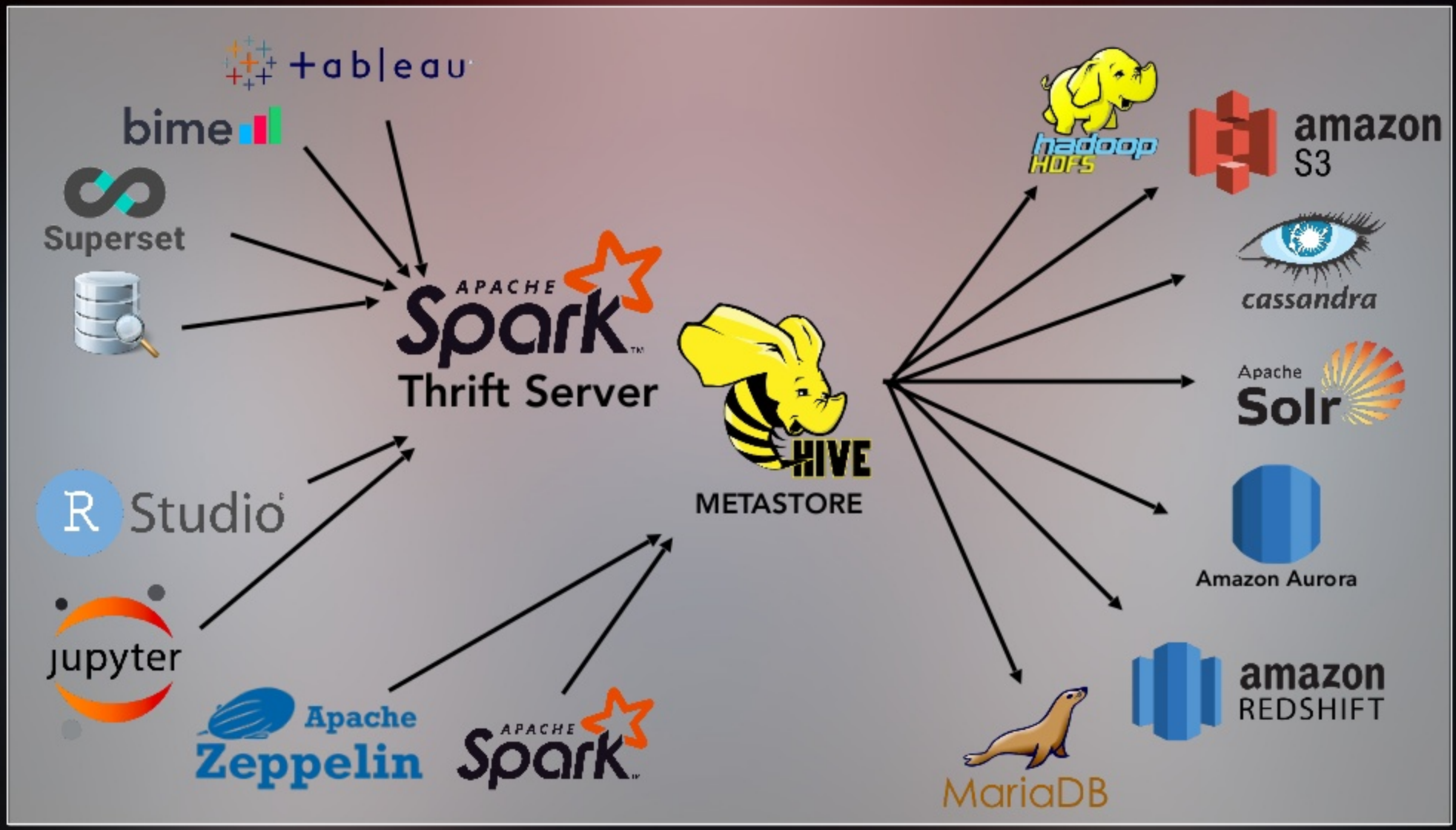

INGEST
Data Ingestion
Storage
PROCESS
Stream
Batch
DISCOVER
Query
Data exploitation
Orchestration

DATA EXPLOITATION

OUR ANALYTICAL STACK

DATA EXPLOITATION

OUR ANALYTICAL STACK

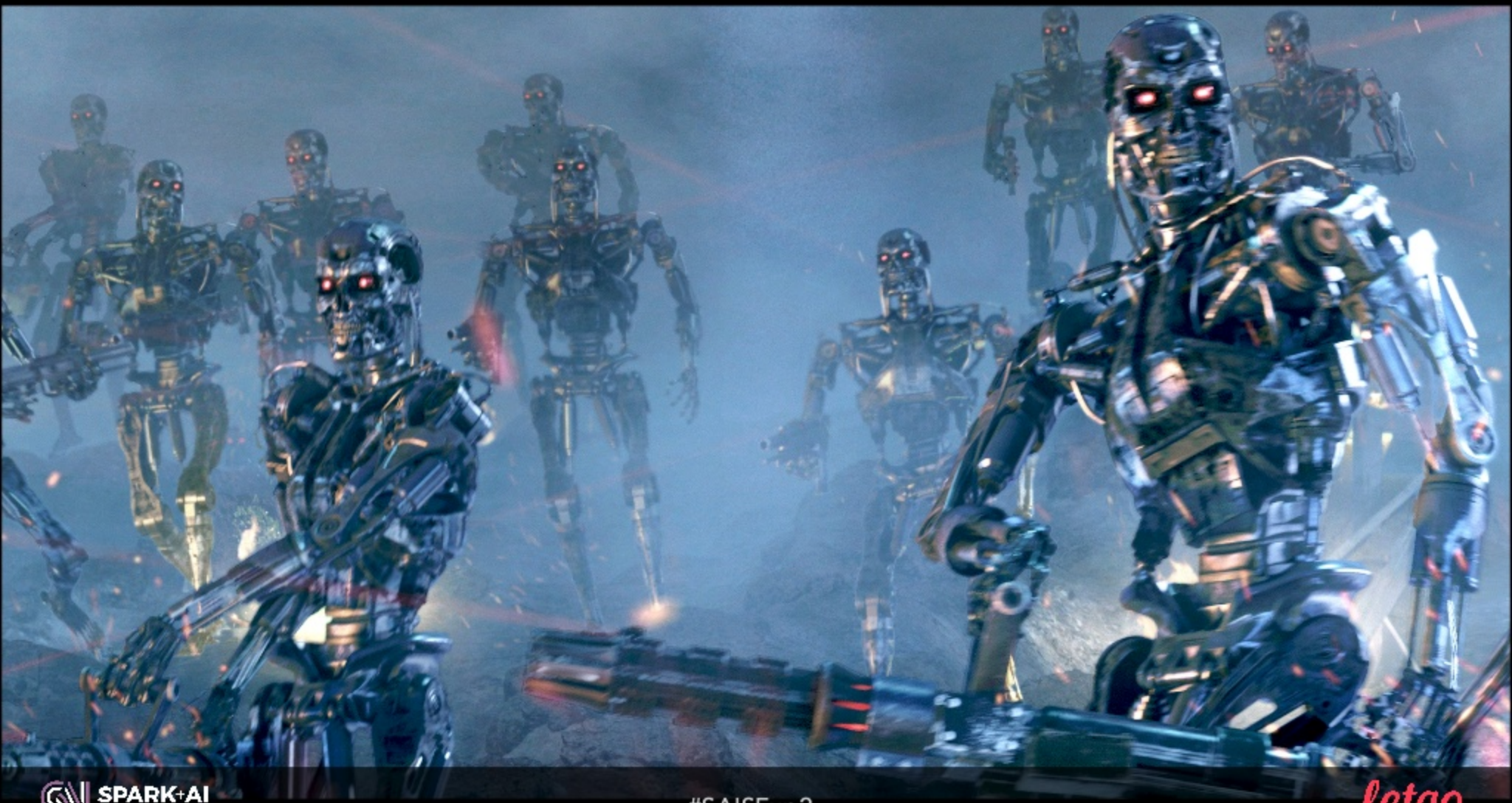


DATA EXPLOITATION

DATA SCIENTISTS TEAM?

DATA EXPLOITATION

DATA SCIENTISTS AS I SEE THEM

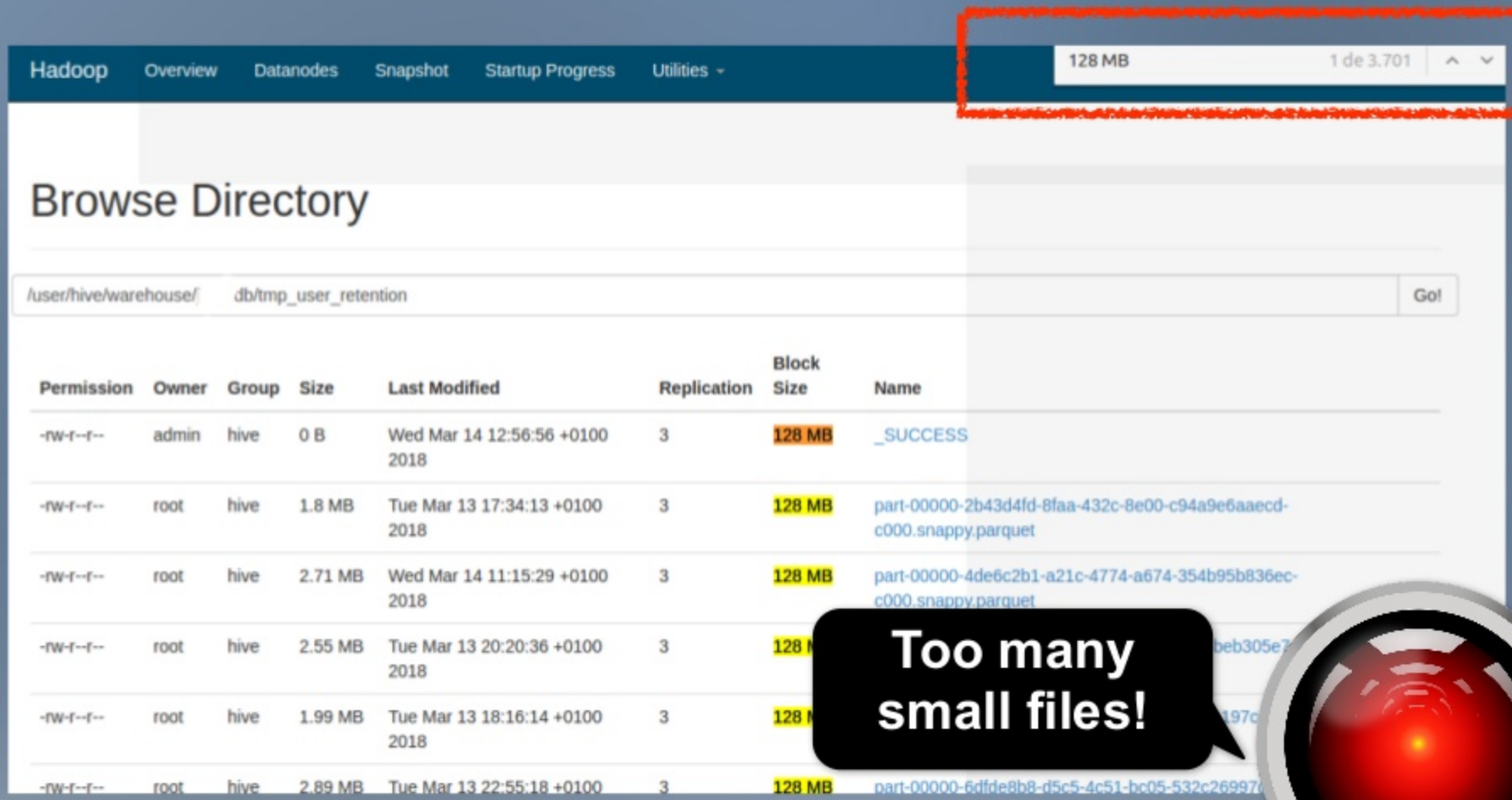


DATA EXPLOITATION

DATA SCIENTISTS SINS

DATA EXPLOITATION

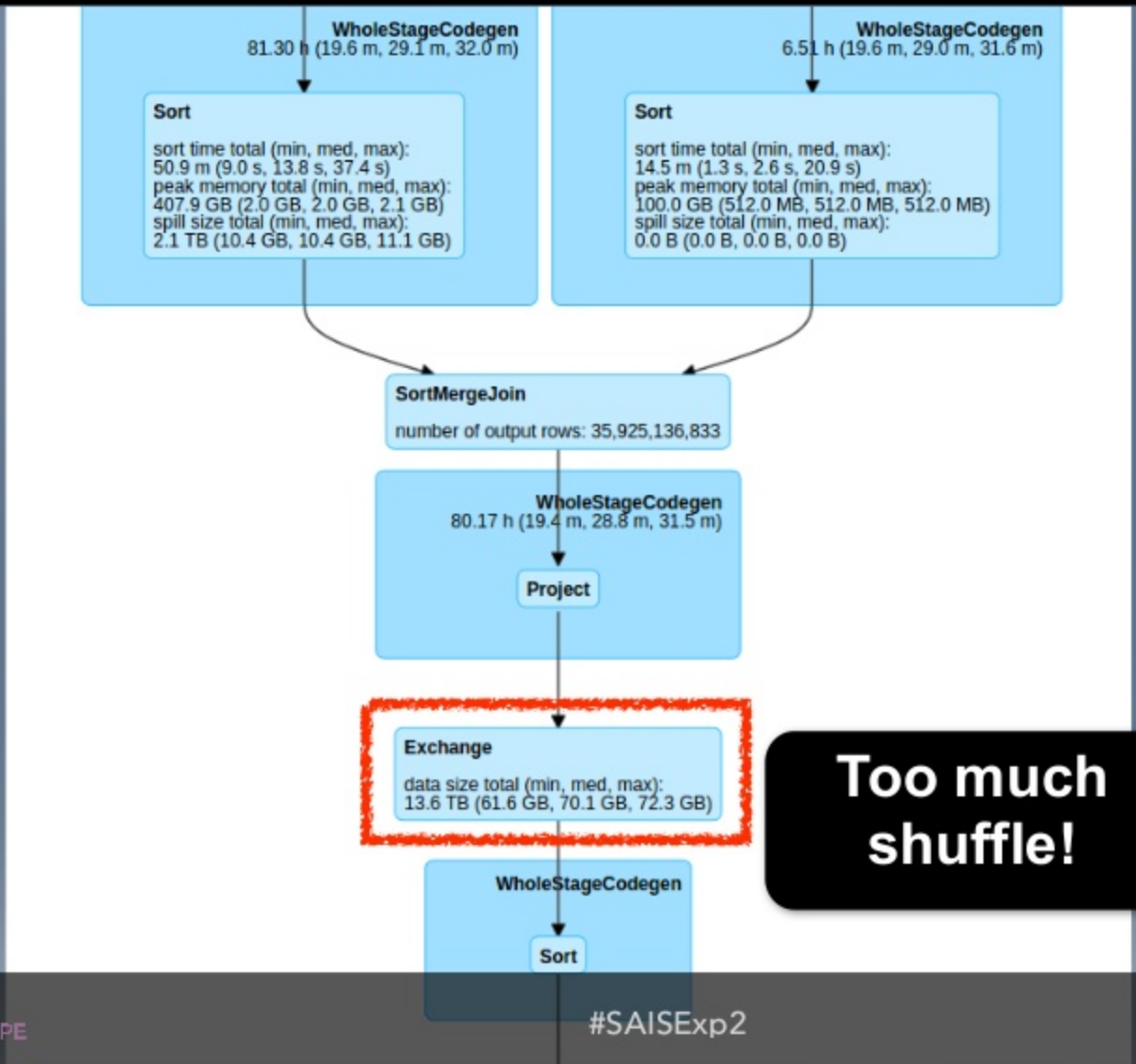
DATA SCIENTISTS SINS

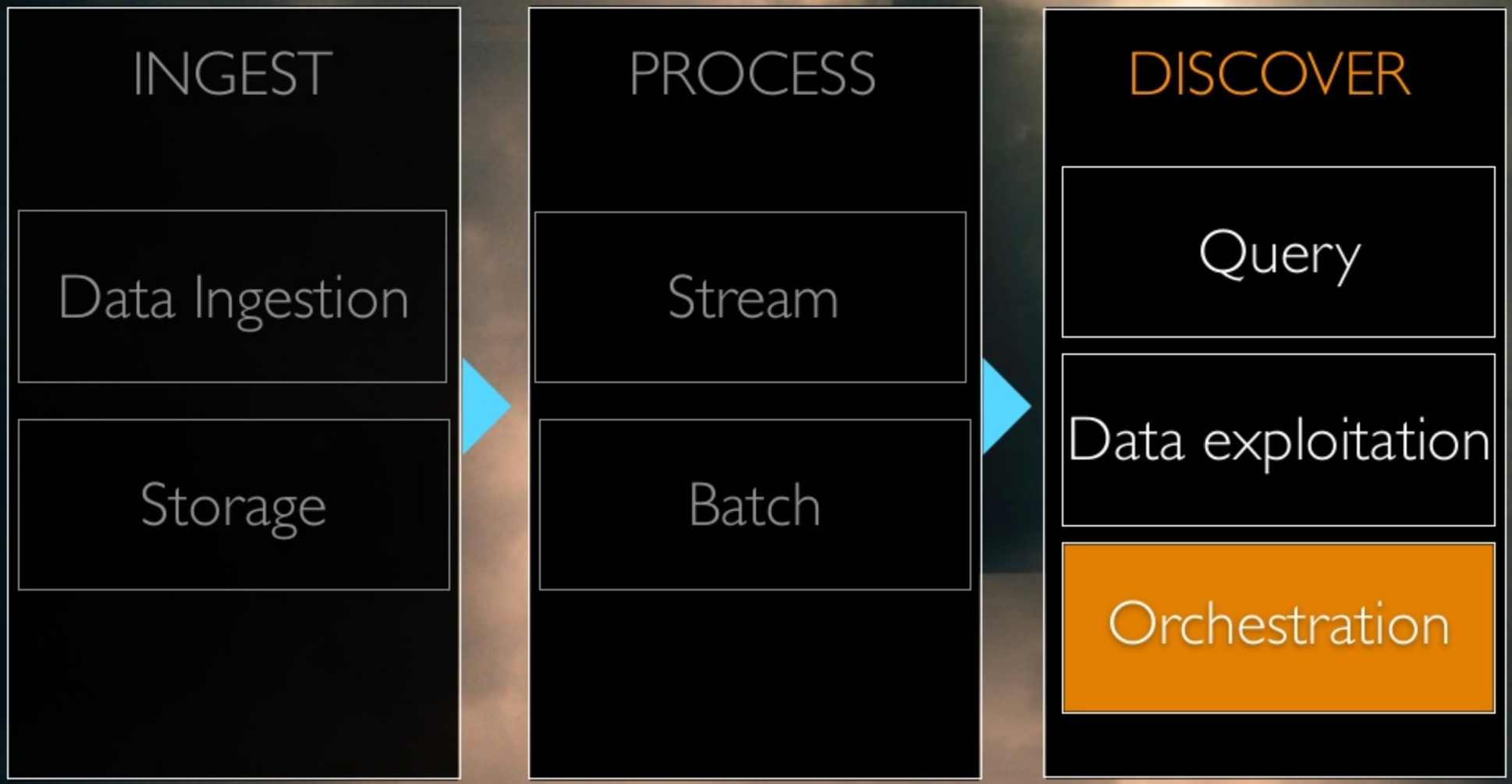


DATA EXPLOITATION

DATA SCIENTISTS SINS

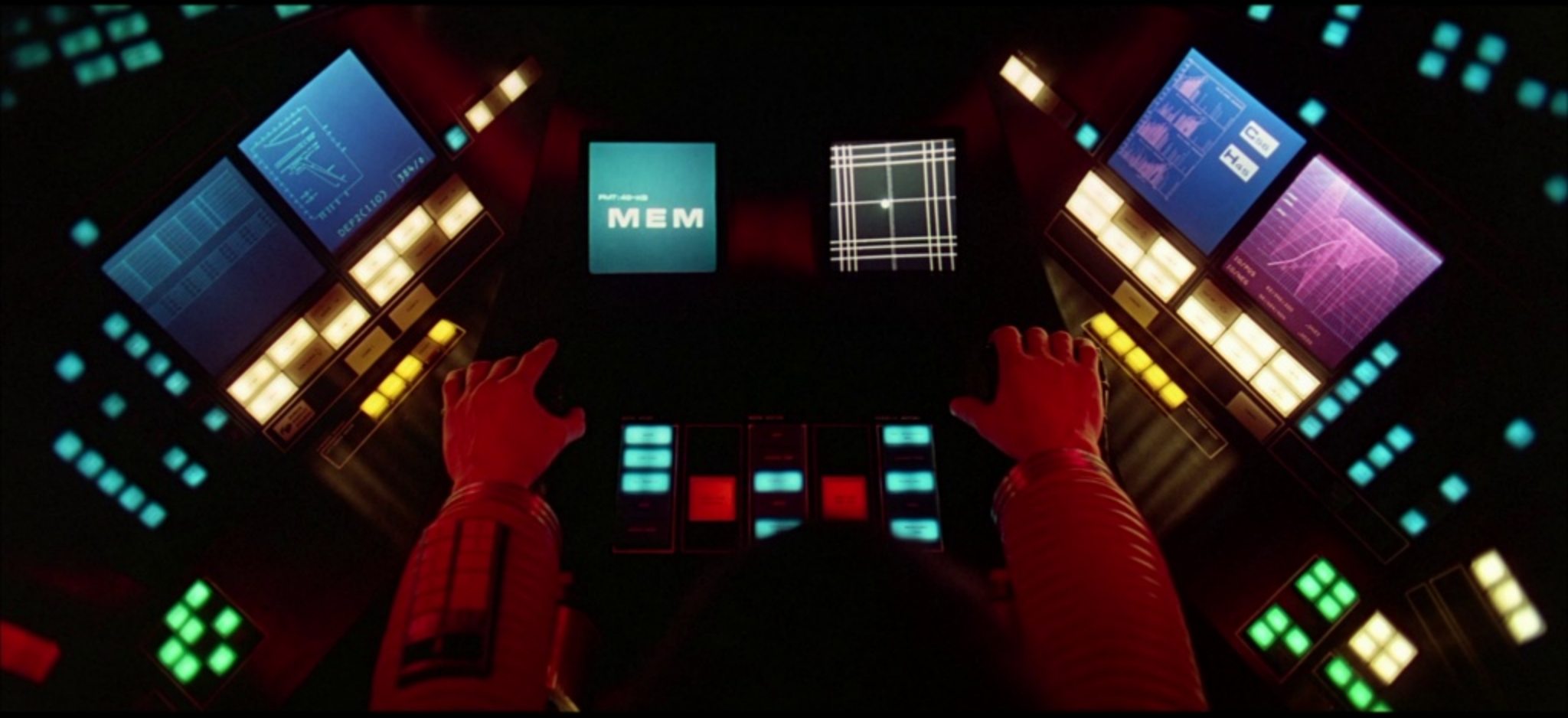
DATA SCIENTISTS SINS




INGEST
Data Ingestion
Storage
PROCESS
Stream
Batch
DISCOVER
Query
Data exploitation
Orchestration

ORCHESTRATION

AIRFLOW



ORCHESTRATION

AIRFLOW

ORCHESTRATION

AIRFLOW

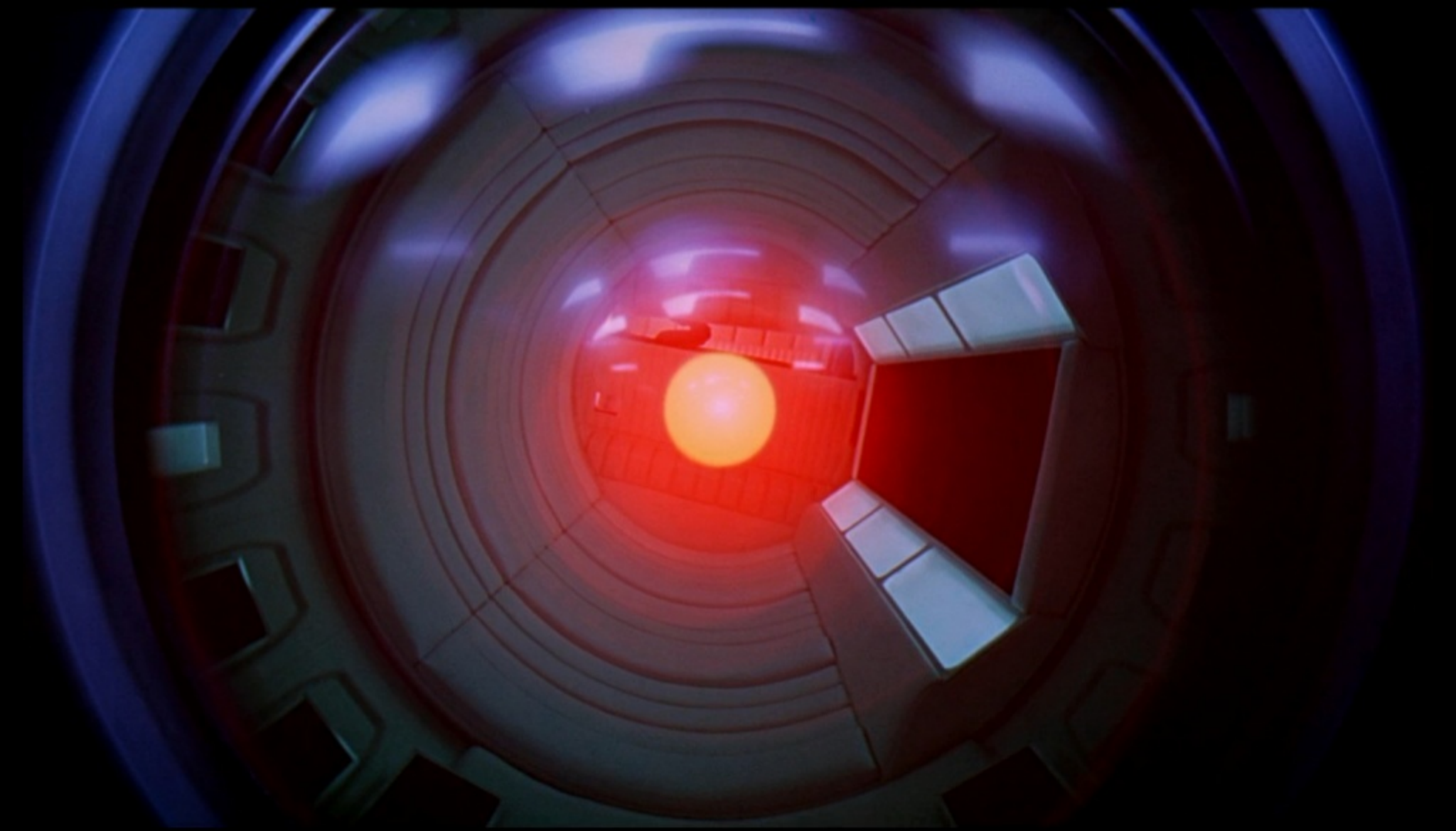
Airflow DAGs Data Profiling Browse Admin Docs About

```
)

create_table_user_skwid_distribution = HiveOperatorLetgo(
    task_id='create_table_user_skwid_distribution',
    dag=dag,
    sql="personalization/create_table_user_skwid_distribution.sql",
    parameters={
        's3_location': path.join(location, 'user_skwid_distribution')
    }
)

insert_into_recently_active_users = HiveOperatorLetgo(
    task_id='insert_into_recently_active_users',
    dag=dag,
    sql="personalization/insert_into_recently_active_users.sql",
    parameters={
        'current_ts': "{{ ts_nodash }}",
        'lookback': lookback
    }
)
```

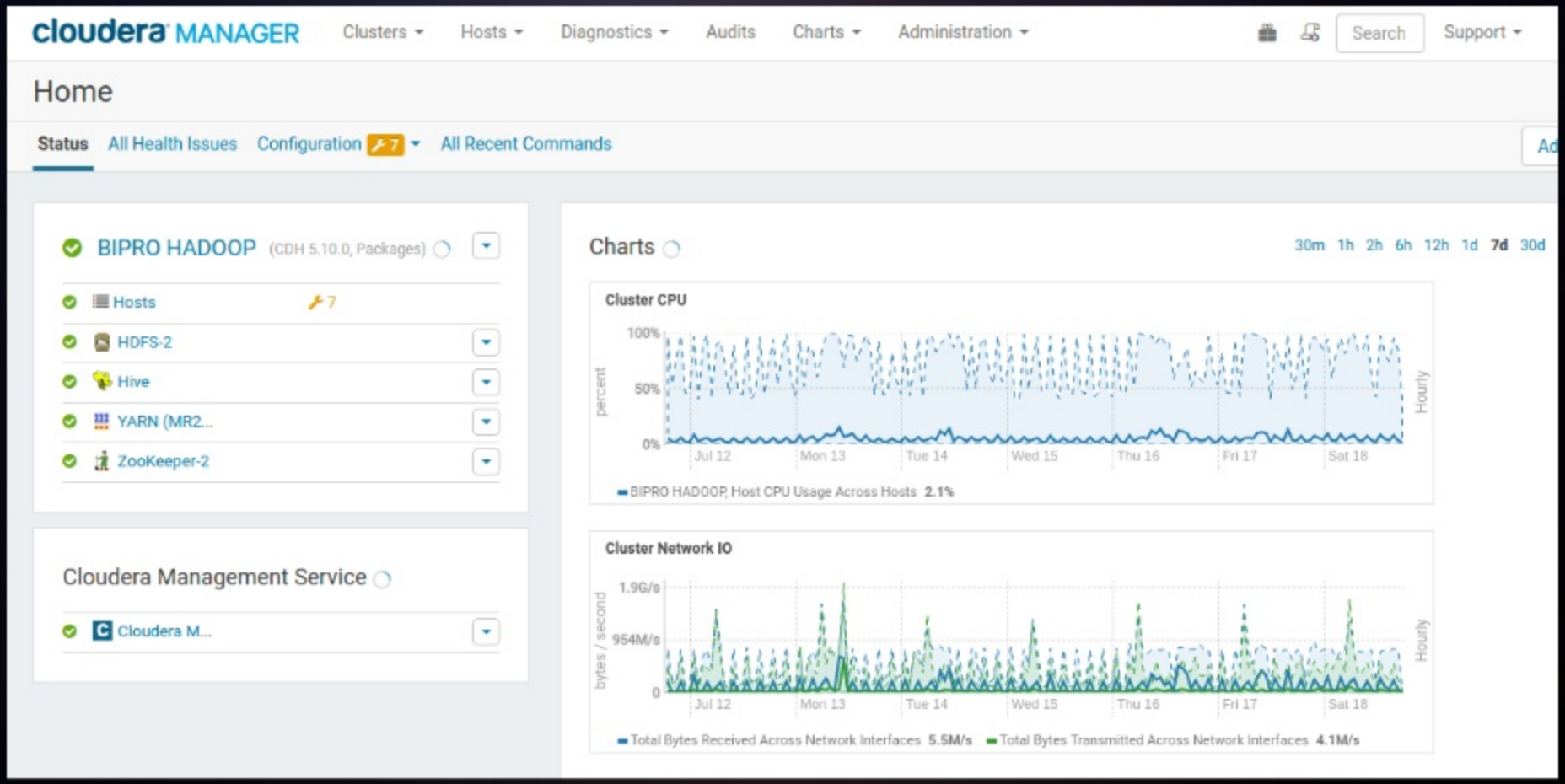

MOVING TO STATELESS
CLUSTER



I'M SORRY RICARDO. I'M AFRAID I CAN'T DO THAT.

MOVING TO STATELESS CLUSTER

PLATFORM LIMITATIONS



MOVING TO STATELESS CLUSTER

PLANNING THE SOLUTION

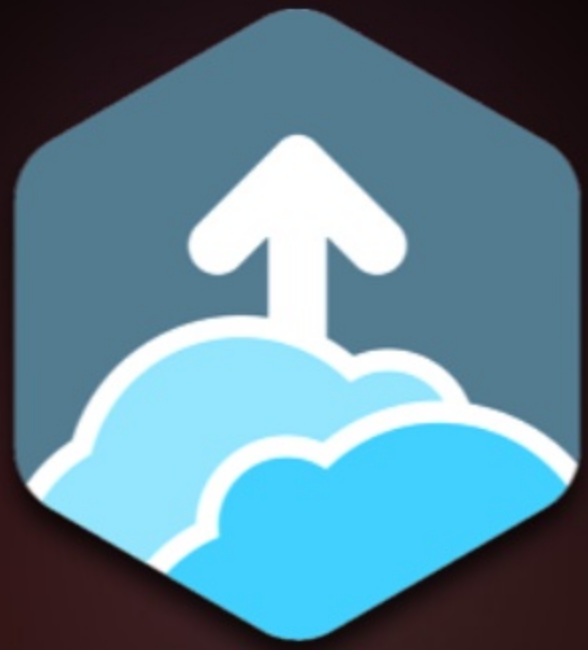
MOVING TO STATELESS CLUSTER

PLANNING THE SOLUTION

amazon
S3

MOVING TO STATELESS CLUSTER

PLANNING THE SOLUTION



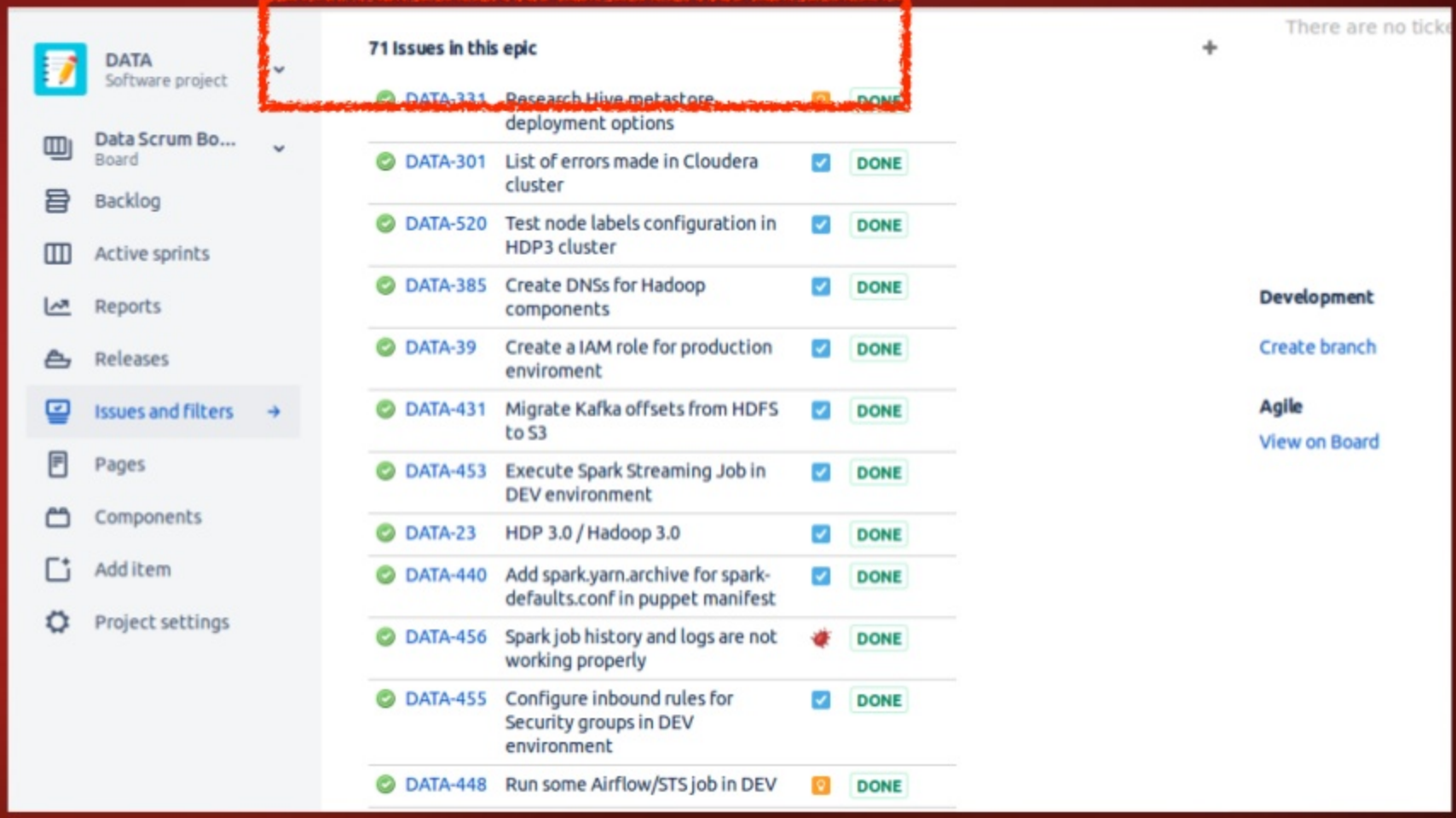
Cloudbreak

MOVING TO STATELESS CLUSTER

A LONG AND WINDING PROCESS

MOVING TO STATELESS CLUSTER

A LONG AND WINDING PROCESS

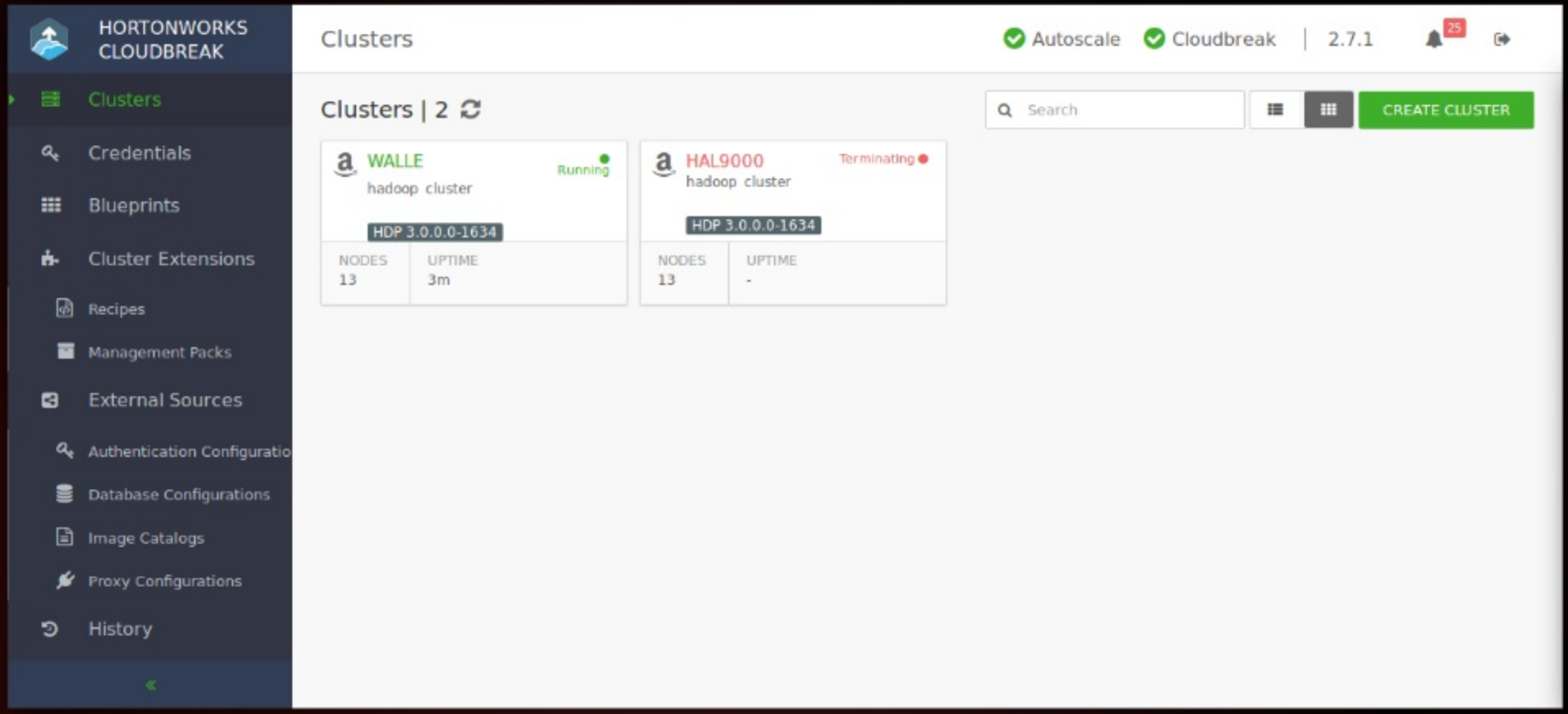


MOVING TO STATELESS CLUSTER

NEW CAPABILITIES OF THE PLATFORM

MOVING TO STATELESS CLUSTER

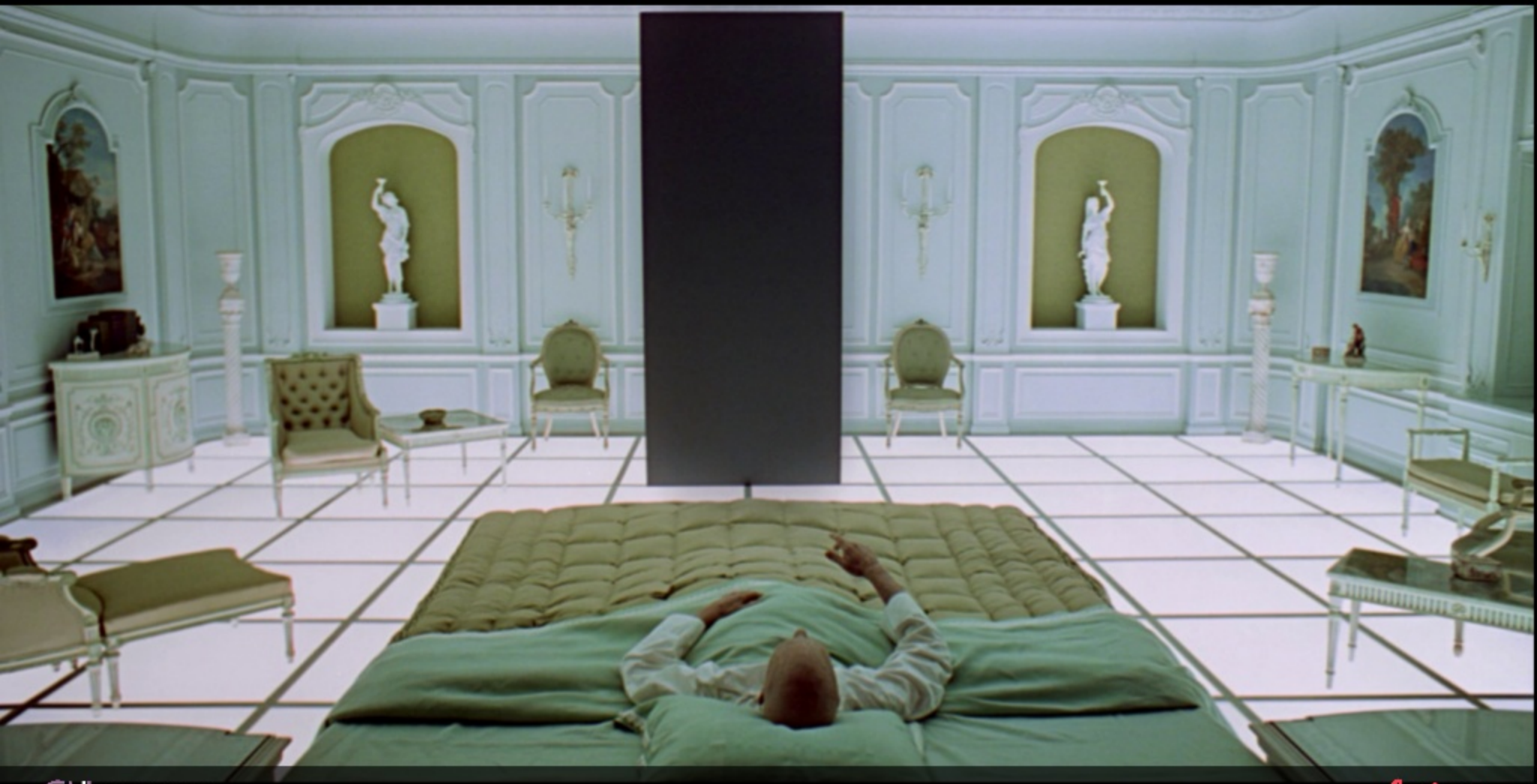
NEW CAPABILITIES OF THE PLATFORM




JUPITER AND BEYOND
THE INFINITE

JUPITER AND BEYOND THE INFINITE

SOMETHING WONDERFUL WILL HAPPEN




presents
I EVE
https://youtu.be/ClnMDMuSFwc

"The only way to discover the limits of the possible is to go beyond them into the impossible."

–ARTHUR C. CLARKE

THE FUTURE...?

DO YOU WANT TO JOIN US?

https://boards.greenhouse.io/letgo