DATA+AI SUMMIT 2022

Powering Up The Business with a Lakehouse

The journey towards data democratization

Ricardo Wagenmaker
Senior Data Engineer, Wehkamp

ORGANIZED BY databricks

Who am I

- Portuguese & Dutch 🇵🇹🇳🇱
- Korfbal player
- NBA and golf
- Wehkamp is one of the biggest online retailers in the Netherlands

wehkamp

Wehkamp

>300.000
different products

>600.000
daily visitors

729,5 million
sales 20/21

75%
of all customers shop via mobile

>1.000
colleagues

11 million
packages sent each year

72%
of our customers are female

Over 2.500 brands
WE Fashion // Vingino //Mango // Tommy Hilfiger // Scotch & Soda // ONLY // Private Label
wehkamp home // HK living // Woood // Zuiver // Riverdale // House Doctor

Agenda

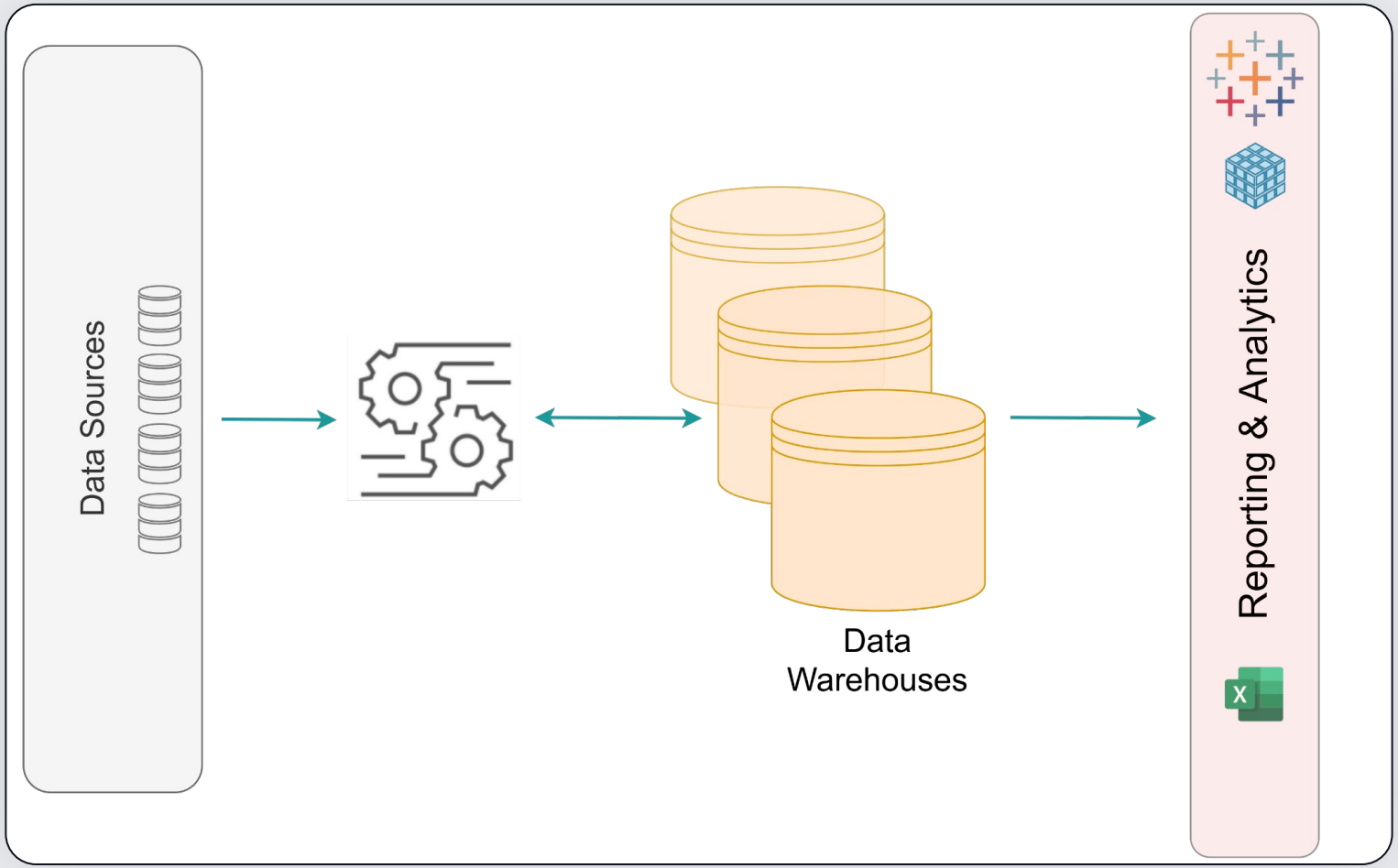
- Where we started
- Lakehouse architecture @ Wehkamp
- Vent-Ingest: Ingestion Framework
- Pseudonymization
- Alerting -> Slack

Where we started

The journey

Traditional BI environment with DWs

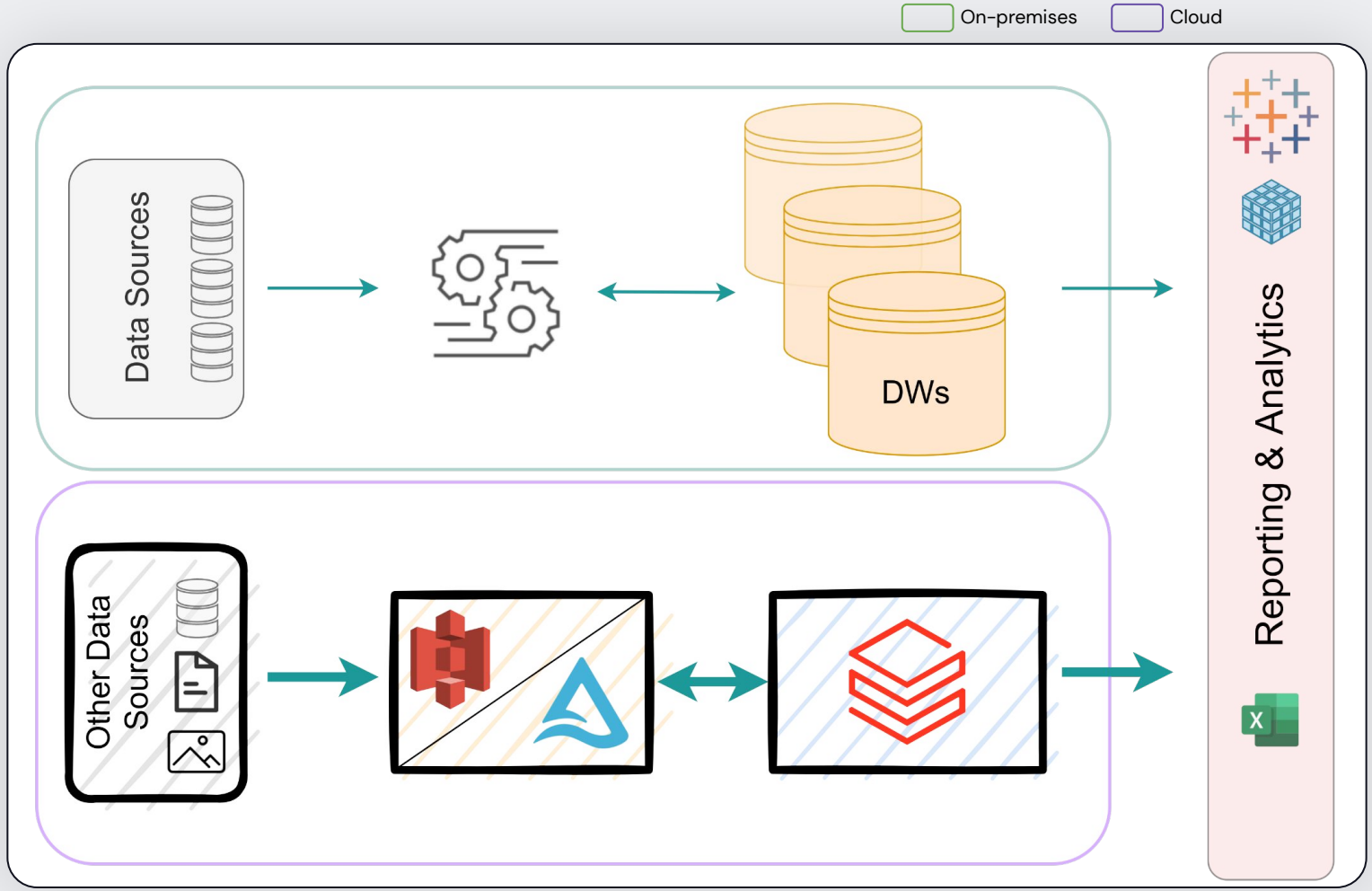
- Stable & high quality environment
- Single source of truth
- Limited to structured data
- On Premise environment



The journey

Databricks gets introduced in the company

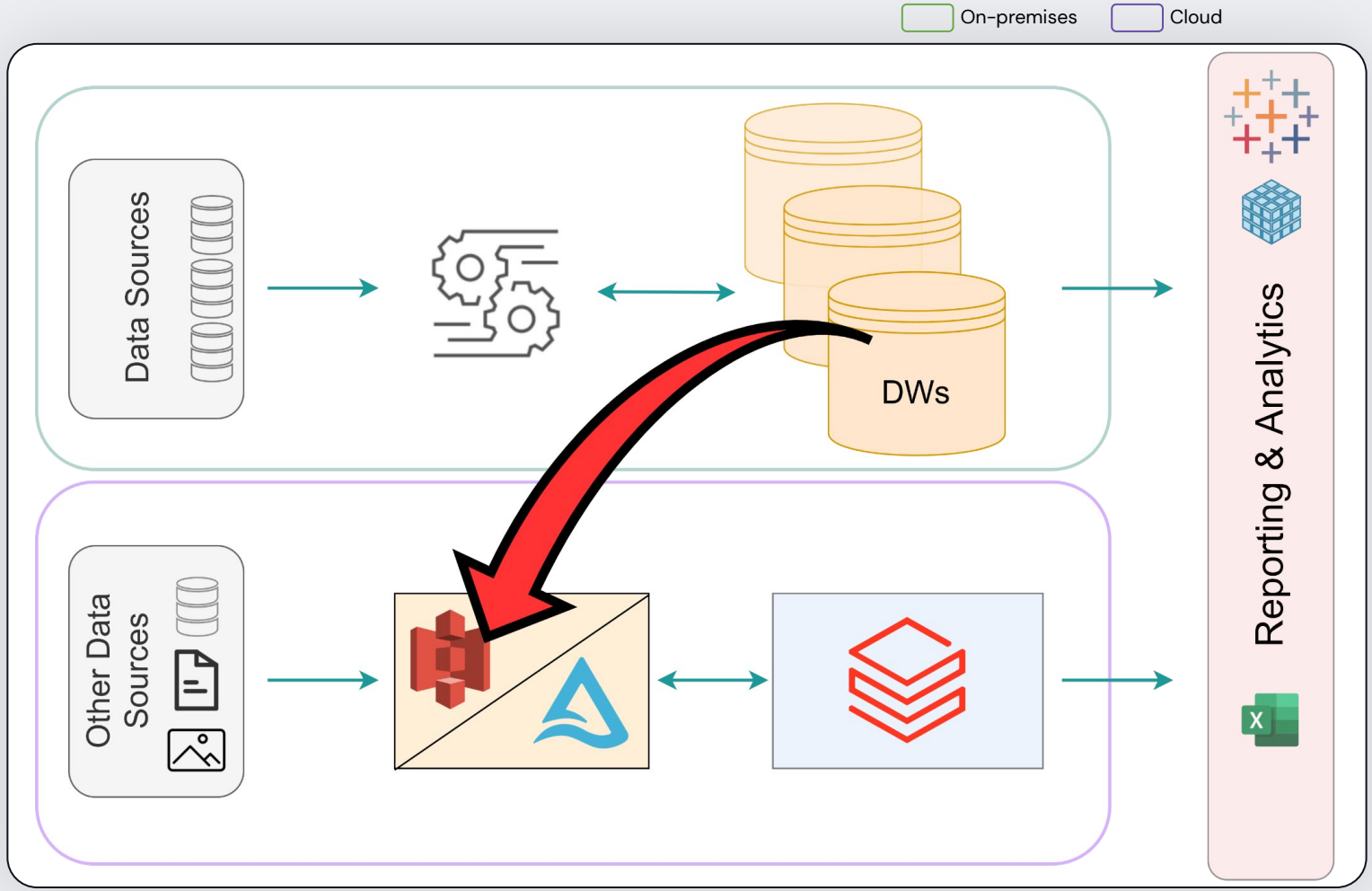
- New uses cases are unlocked
- New data sources are added to S3
- Self service environment for data exploration



The journey

Old data. New world.

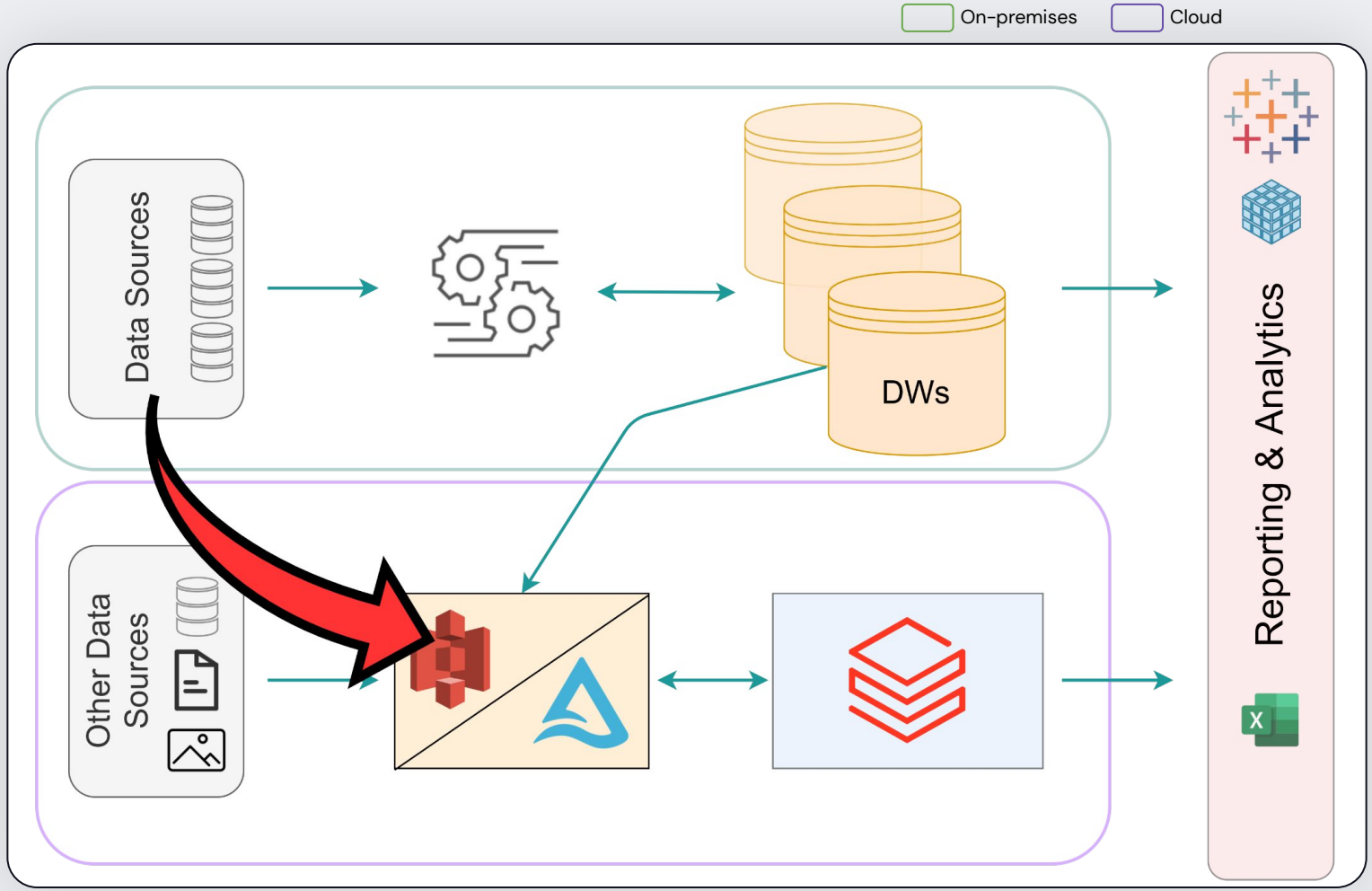
- Good quality data joins distributed computing
- More power to data users, same old trustworthy data



The journey

Same data… New routes?

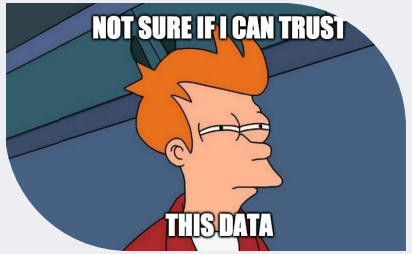
- Keeping our DWs in the pipeline was becoming a bottleneck for some use cases
- Shortcuts and temporary implementations were made
- Speed ↗
  Quality ?
  Stability ↘

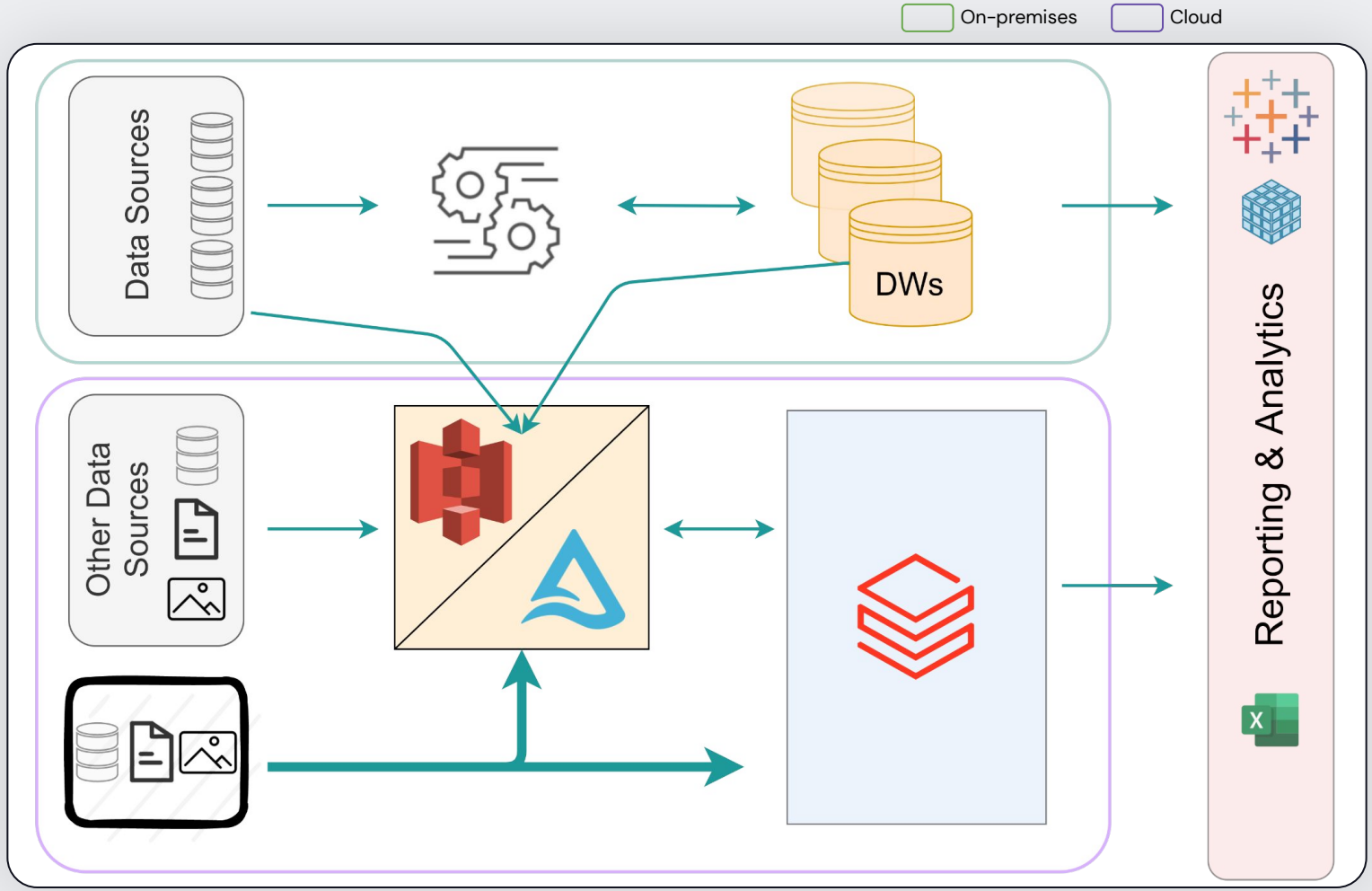


The journey

Even more data

- Wider adoption meant even more data sources and even more teams starting to create their own independent pipelines

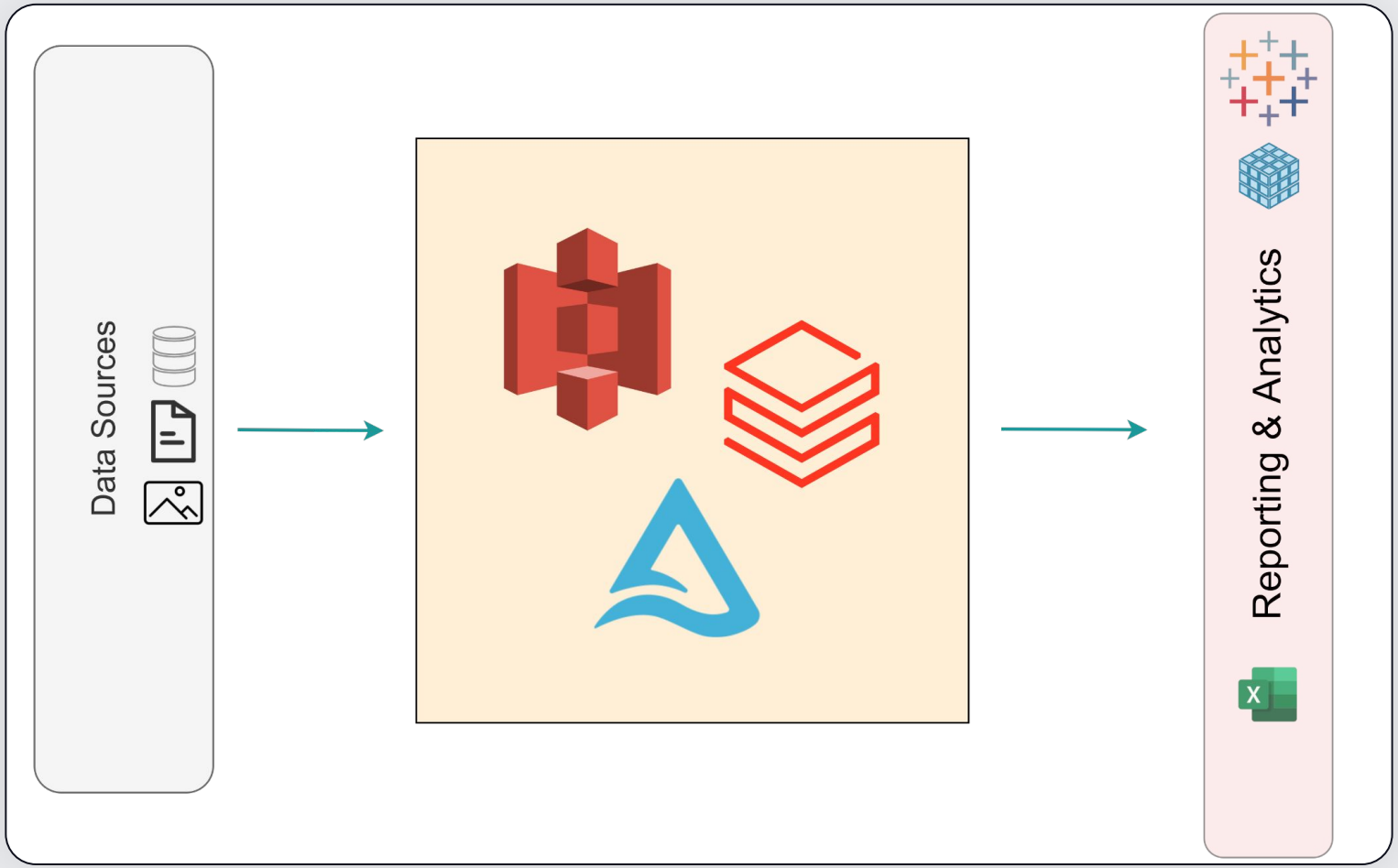




The journey

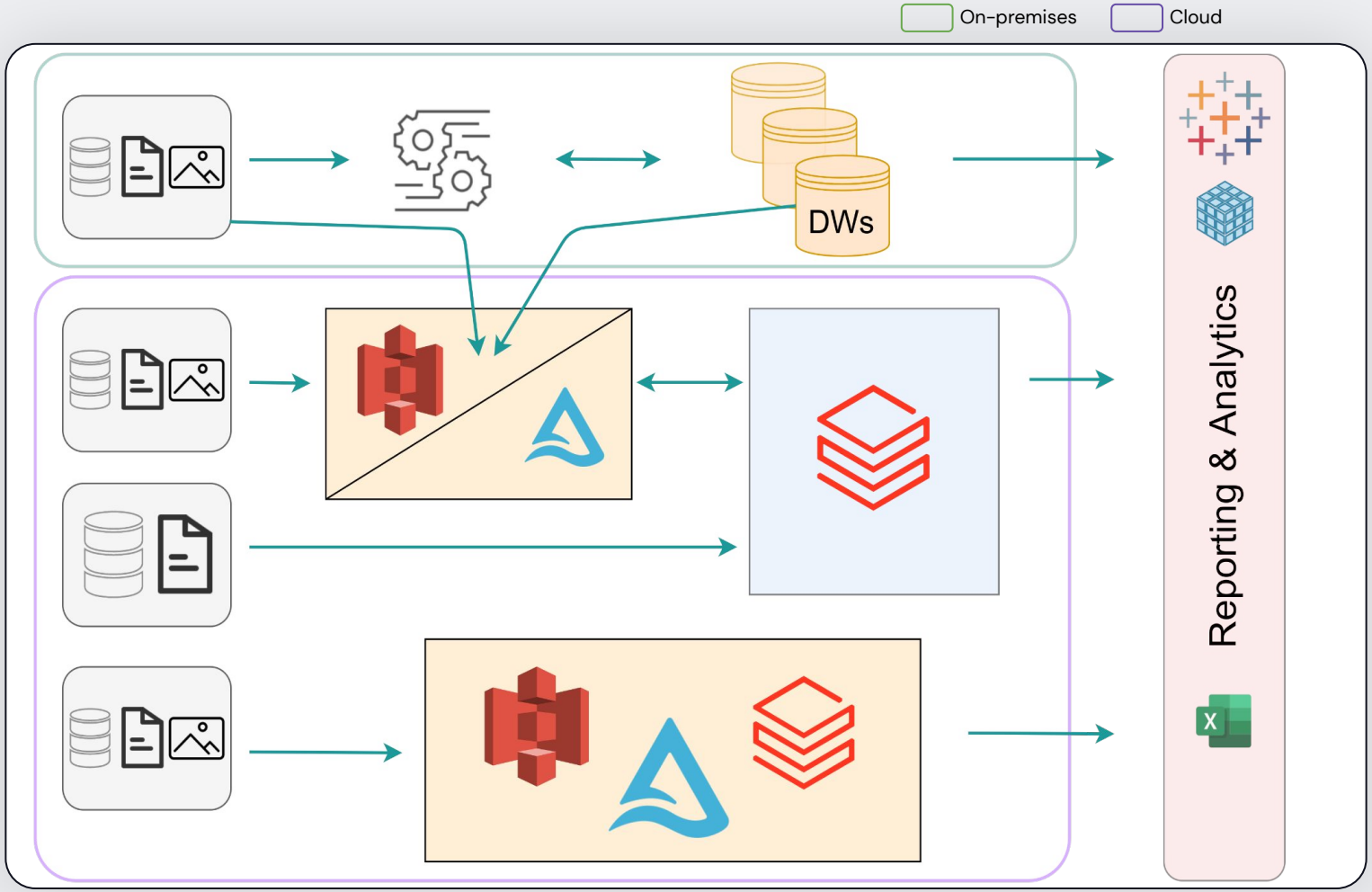
Endstation: The Lakehouse.

- Lakehouse powered by a Delta Lake
- Unified and governed data usage
- Simplified architecture



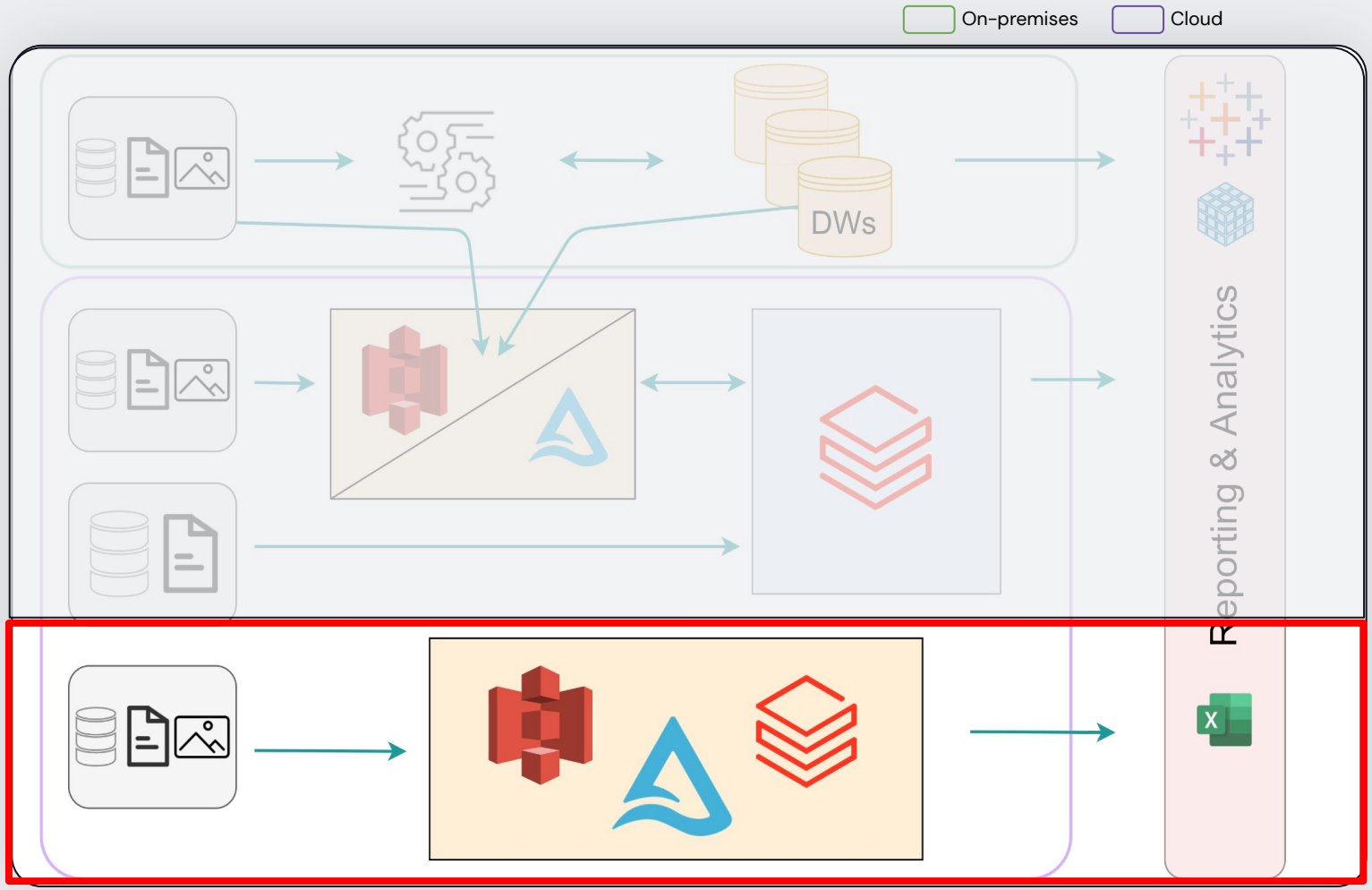
The journey

The honest lens



The journey

The honest lens



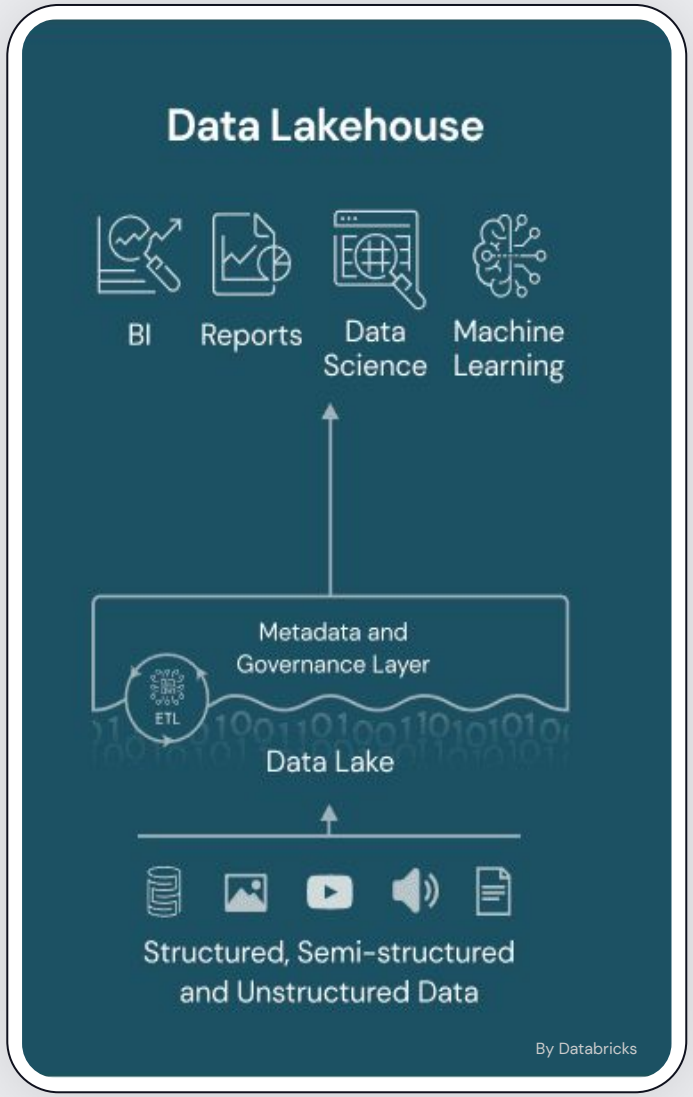
Lakehouse

Data Warehouses

High performance

Data Lakes

Scalability, cost-efficiency, and flexibility



Lakehouse Architecture @ Wehkamp

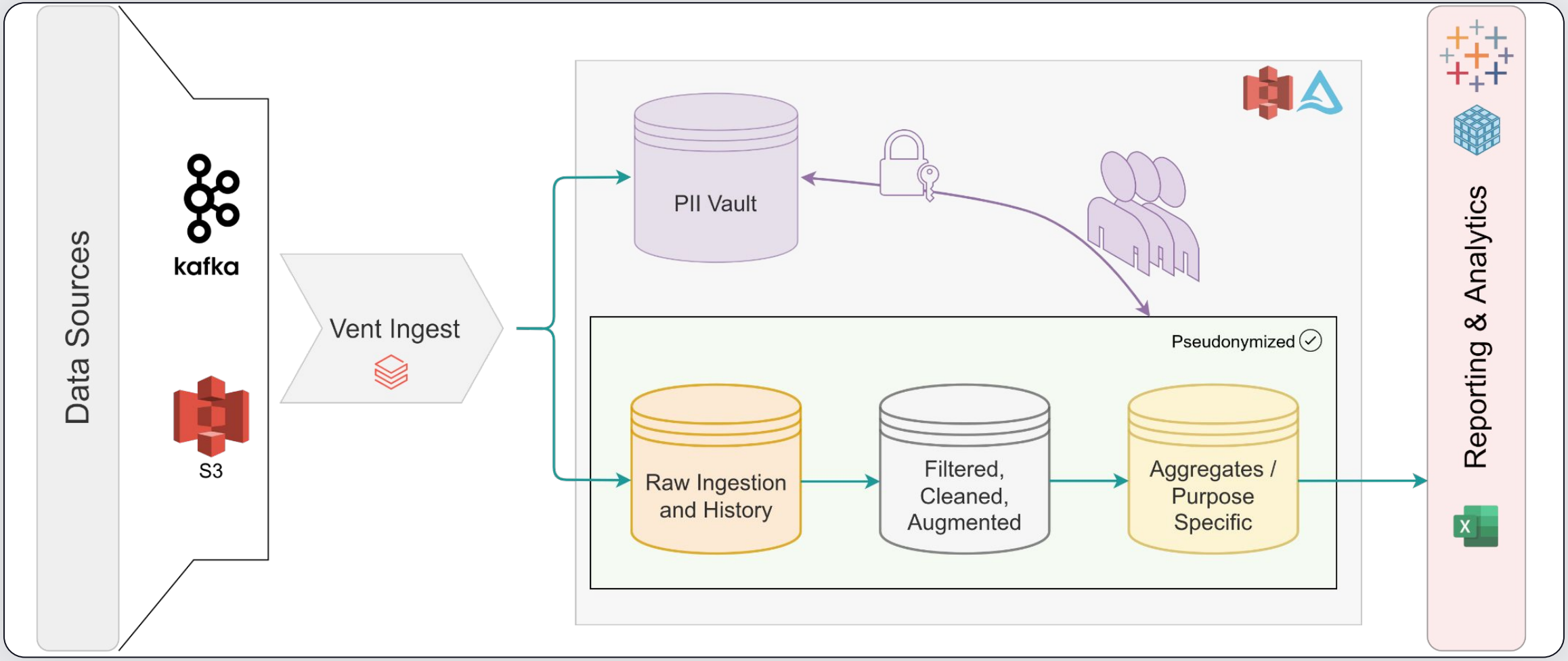
Lakehouse Architecture @ Wehkamp

Unifying the data access

- Delta Lake as storage layer
- Incremental ingestion
- Incremental processing with the Medallion Architecture: Bronze -> Silver -> Gold
- All PII data in the Delta Lake should be pseudonymised
- Column naming standards
- Easy way to let data users apply the best practices

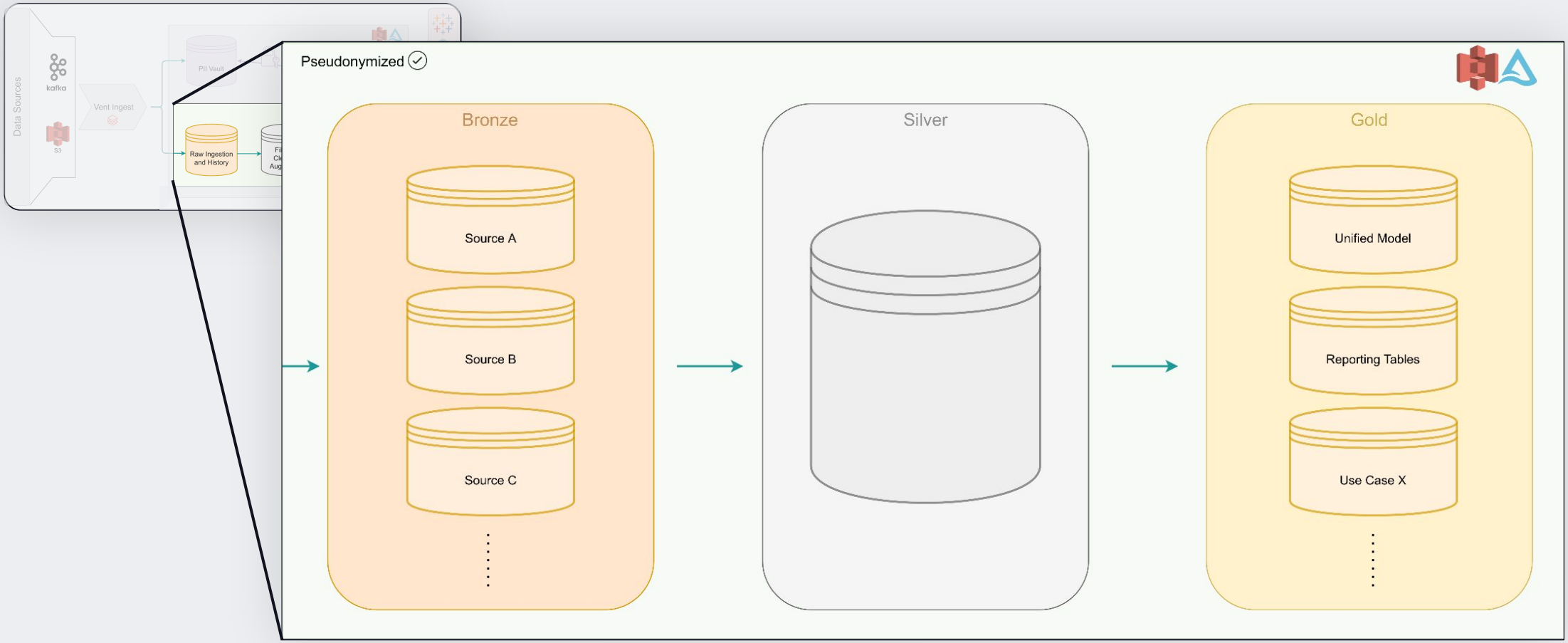
Lakehouse Architecture @ Wehkamp

Unifying the data access



Lakehouse Architecture @ Wehkamp

Inside the delta lake

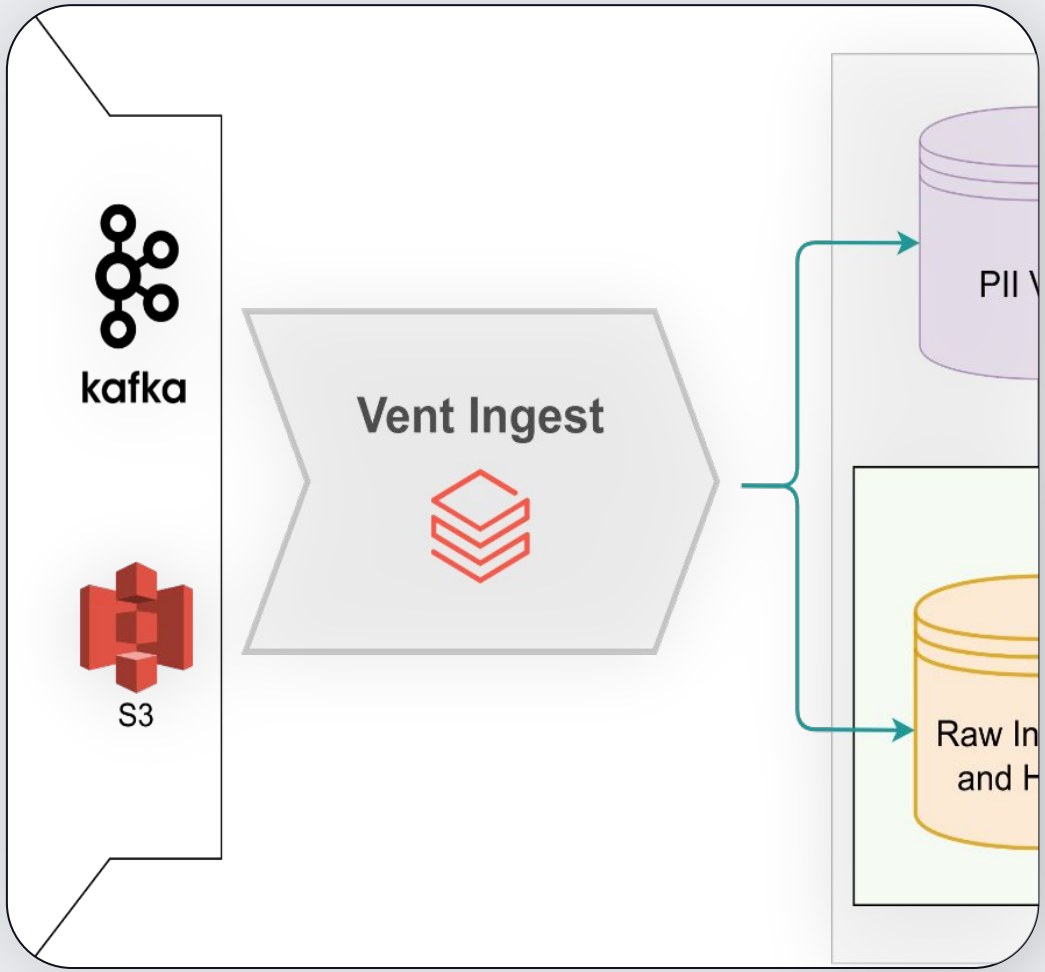


Vent-Ingest

Vent Ingest

Idea And Design: Framework + Library

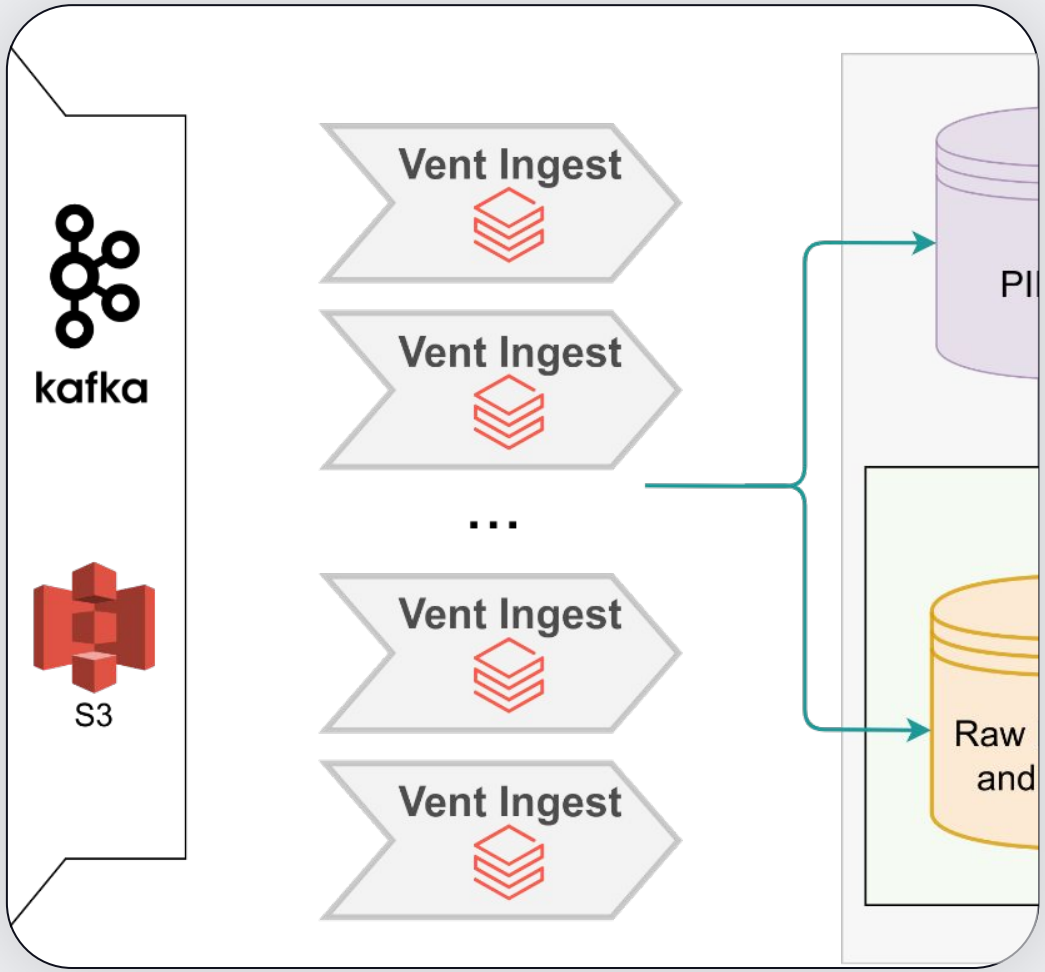
- Custom python library (*.whl*)
- Modular Pyspark code with VSC
- Easy ingestion via JSON config files
- Pseudonymization of all PII fields
- Stream ingestion with Spark's Structured Streaming API
  - Current support: Kafka & S3 + autoloader
  - Batch is supported through Trigger.Once



Vent Ingest

How does it run

- Each new ingestion source is defined by 2 sets of configurations
  - Read options
  - Data specific + Write options
- Several jobs run independently and at a different frequency. Each one ingests the sources that are defined to run at that frequency.
  - Parallel execution within the job via a spark scheduler pool



Vent Ingest

connections.json

1

```
1  {
2   "$schema": "../schemas/config-schema.json",
3   "connections": [
4        {
5         "name": "awesome-bucket",
6         "url": "dbfs:/mnt/awesome-bucket",
7         "format": "autoloader-json",
8         "options": {
9           "cloudFiles.format": "json",
10          "cloudFiles.region": "eu-west-1",
11          "cloudFiles.includeExistingFiles": "true",
12          "cloudFiles.useNotifications": "true",
13          "cloudFiles.triggerOnceQueueFlushTimeout": "5s"
14        }
15      ]
16 }
```

Vent Ingest

wishlist.json

2

```
1  {
2   "$schema": "../../schemas/table.schema.json",
3   "table": "wishlist",
4   "database": "dl_brz_sourceA",
5   "connection": "awesome-bucket",
6   "folder": "wishlist",
7   "frequency": "realstream",
8   "retention": "7 years",
9   "mode": "append",
10  "comment": "JSON | Data from wishlist service",
11  "trigger_options": { "processingTime": "1 minute" },
12  "partition_by": [ "dateAdded" ],
13  "fields": [
14    {
15      "metadata": {"pii_field": "customer_number"},
16      "name":"customerNumber",
17      "nullable":true,
18      "type":"string"
19    },
20    {
21      "metadata": {},
22      "name": "dateAdded",
23      "nullable": true,
24      "type": "string"
25    },
26    {
27      "metadata": {},
28      "name": "priceWhenAdded",
29      "nullable": true,
30      "type": "long"
31    },
32    {
33      "metadata": {},
34      "name": "productNumber",
35      "nullable": true,
36      "type": "string"
37    },
38    {
39      "metadata": {
40        "comment": "Size specification"
41       },
42      "name": "sizeCode",
43      "nullable": true,
44      "type": "string"
45    },
46  ]
47 }
```

Vent Ingest

In action

- In place transformation for flat and nested schemas
- Using explicit schemas means that schema evolution needs to be handled by the owner of such config

(1) Spark Jobs

wishlist_brz: pyspark.sql.dataframe.DataFrame

```
customerNumber: string
dateAdded: string
priceWhenAdded: long
productNumber: string
sizeCode: string
```

Table | Data Profile

	customerNumber	dateAdded	priceWhenAdded	productNumber	sizeCode
1	IcEjdsdEPVs3r7q3nXA5DGhEn3M	2022-06-06T10:57:14.789Z	449	16951228	null
2	I3INcb1wNpygPG4Tj3jojzEAUR0	2022-06-06T10:57:13.799Z	2999	16939486	null
3	O005ixrCa1vsbZGWNBsBs48ZwfI	2022-06-06T10:57:13.760Z	2499	16856791	null
4	xmpyYOPSLRu07qbb3uOAPUrnP5g	2022-06-06T10:57:12.642Z	9995	16651752	null
5	V7s0bpSjrSvmc5aG3GYtG$dD6Uc	2022-06-06T10:57:12.378Z	2449	16875658	null
6	9cUzfeWdQCQdjsz8p9QUNkp4AJs	2022-06-06T10:57:11.436Z	4000	16916202	null
7	04wofvKzaet39mmO7v5jf7EA4XE	2022-06-06T10:57:10.875Z	3149	16924140	null
8	$mL6PGxho8bWdZI_f8OBkeL$xY4	2022-06-06T10:57:10.594Z	2999	16921314	null
9	nN6WaaHnN0Sv7fYTsTMkIMvswXw	2022-06-06T10:57:10.453Z	3999	16887466	null
10	XQAhauge7NOrSy4iUR09HcRJj7A	2022-06-06T10:57:10.256Z	3745	16576854	null
11	qDEfntqTJMvV7tiwL m2kXOiLikk	2022-06-06T10:57:09.831Z	6500	16867408	null

Truncated results, showing first 1000 rows.

Pseudonymization

Pseudonymization

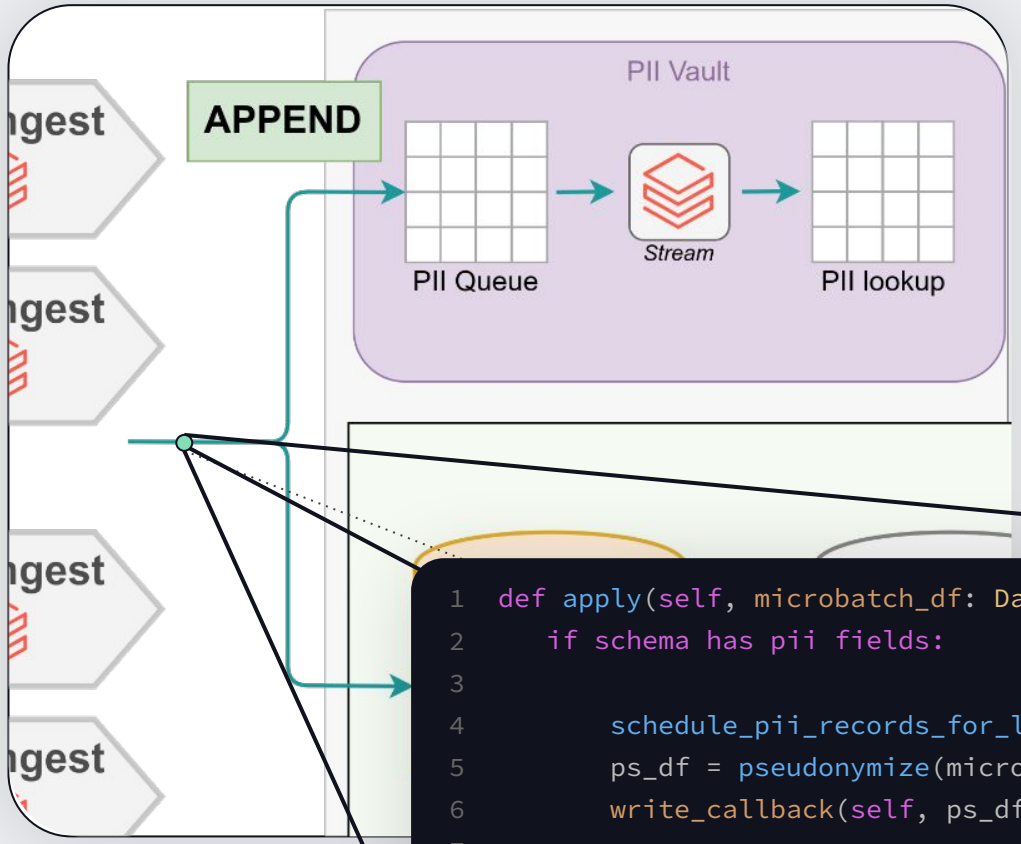
The logic behind it

- Democratize data access by restricting access to PII data to specific purposes
- One-way cryptographic hashing and salting
- Use the concept of PII types to differentiate the purposes for accessing PII data.
  a. *E.g. customer_number, customer_email, employee_name, etc*
- Encode the hashes with base64 for storage efficiency

```
def pseudonymization_expression(pii_type, column) -> str:
    normalized = f"Normalize in lowercase the {column} as a string"
    sha = f"sha1({pii_type} + secure_salt + {normalized}))"
    # save storage space
    return f"translate hexadecimal encoding to base64({sha})"
```

Pseudonymization

Updating the lookup table




```python
def apply(self, microbatch_df: DataFrame, write_callback):
    if schema has pii fields:

        schedule_pii_records_for_lookup(microbatch_df, self)
        ps_df = pseudonymize(microbatch_df, self)
        write_callback(self, ps_df)

    else:
        write_callback(self, microbatch_df)
```

- All streams write concurrently to the PII Queue table with append mode
- A separate job updates the PII lookup table

Alerting:
Runtime & Quality -> Slack

Alerting -> Slack

Keeping the incidents where people are

- Slack at Wehkamp is crucial for incident management
- For managing the delta lake we have 2 channels
  - Runtime alerts
  - Quality alerts via PyDeequ
- Slack Webhook + Databricks secrets
- Alerting on table level instead on job level

Alerting -> Slack

Keeping the incidents where people are

Apr 27th

LakehouseSentinel APP 06:43

Error on job: Vent Ingest 10 min Schedule, run: 230598491

Error updating table `dl_brz_trc.trc_orderregels`

bronze

1 reply

Thread

Message in #vent-ingest-alerts

LakehouseSentinel APP
Apr 27th at 06:43

Error on job: Vent Ingest 10 min Schedule, run: 230598491

Error updating table `dl_brz_trc.trc_orderregels`

bronze

1 reply

LakehouseSentinel APP 06:43

```
Traceback (most recent call last):
```

LakehouseSentinel APP Apr 27th at 06:43
dl_brz_trc.trc_orderregels

```
    return_value = get_return_value(
  File "/databricks/spark/python/pyspark/sql/utils
    raise converted from None
pyspark.sql.utils.StreamingQueryException: An exce
  File "/databricks/spark/python/lib/py4j-0.10.9.1
    return_value = getattr(self.pool[obj_id], meth
  File "/databricks/spark/python/pyspark/sql/utils
    raise e
  File "/databricks/spark/python/pyspark/sql/utils
    self.func(DataFrame(jdf, self.sql_ctx), batch_
  File "/databricks/python/lib/python3.8/site-pack
    raise e
  File "/databricks/python/lib/python3.8/site-pack
    return table_source.apply(batch_df, self.write
  File "/databricks/python/lib/python3.8/site-pack
    return write_callback(self, batch_df)
  File "/databricks/python/lib/python3.8/site-pack
    return self.write(table_source, batch_df)
  File "/databricks/python/lib/python3.8/site-pack
    raise e
  File "/databricks/python/lib/python3.8/site-pack
    return self._merge(table_source, df, existing_
  File "/databricks/python/lib/python3.8/site-pack
    (dt.alias('source')
  File "/databricks/spark/python/delta/tables.py",
    self._jbuilder.execute()
  File "/databricks/spark/python/lib/py4j-0.10.9.1
    return_value = get_return_value(
  File "/databricks/spark/python/pyspark/sql/utils
    raise converted from None
delta.exceptions.ConcurrentDeleteDeleteException:
Conflicting commit: {"timestamp":1651034614084,"us
Refer to https://docs.databricks.com/delta/concurr

=== Streaming Query ===
Identifier: ingest-dl_brz_trc.trc_orderregels [id
Current Committed Offsets: {CloudFilesSource[dbfs:
Current Available Offsets: {CloudFilesSource[dbfs:

Current State: ACTIVE
Thread State: RUNNABLE

Logical Plan:
CloudFilesSource[dbfs:/mnt/nl-wehkamp-data-trc/TRC
```

Conclusions and next steps

Looking Back

- Traditional BI -> Lakehouse journey
- 100+ streams with insignificant downtime
- Adoption of the delta lake is steadily increasing
- Good foundation for improving the performance of our older pipelines and facilitate future use cases the business requires

Whats Next?

- Continue to fill the delta lake with the necessary data to make the migration possible
- Continue to iterate and improve the platform itself
- Expand the platform for support multi label
- New kids to the Databricks E2 party. Integrate some of the new features into our Lakehouse toolkit

DATA+AI
SUMMIT 2022

Thank you

Ricardo Wagenmaker
Senior Data Engineer, Wehkamp