Data Engineering

Powered by

Presenter

in /arifmazumder

Mohammed Arif, PhD

Lead Data Scientist

Big Data | Machine Learning | AI

Microsoft CERTIFIED AZURE DATA SCIENTIST ASSOCIATE

Microsoft CERTIFIED AZURE DATA ENGINEER ASSOCIATE

CCA cloudera SPARK & HADOOP DEVELOPER

Mohammed Arif has more than nineteen (19+) years of working experience in Information Communication and Technology (ICT) industry. The highlights of his career are more than seven (7) years of holding various senior management and/or C-Level and had five (5) years of international ICT consultancy exposure in various countries (APAC and Australia), specially on Big Data, Data Engineering, Machine Learning and AI arena.

He is also Certified Trainer for Microsoft & Cloudera.

MICROSOFT CERTIFIED TRAINER Microsoft MCT Trainer 2023 - 2024

Related Resources

https://arif.works/de

Disclaimer

- Data Engineering is a vast domain
- Emphasize on Practical / Industry Implementation
- Take it as Experience Sharing Session

What is data engineering?

SQL	Python	ETL	Data warehouse
Data ingestion	ML/AI	DataOps	Cloud infra
Data modeling	Data quality	Data governance	Security
Data observability	Data versioning	Data cleaning	Optimizations

What is data engineering?

Data → "Flow"

Upstream		Downstream
Data producers →	Data manipulated (Data engineers) →	Data consumers

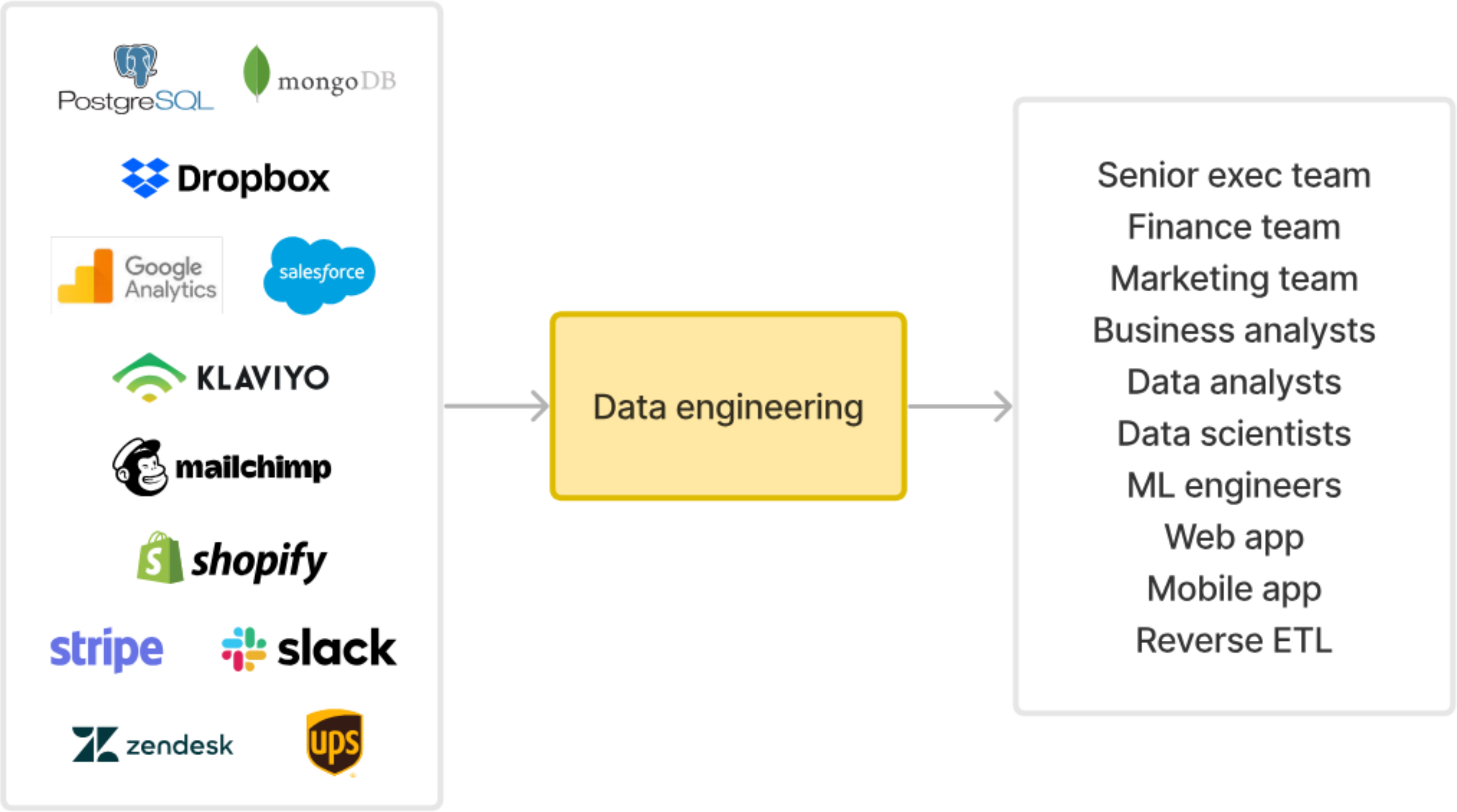
[Website]

order_id	amount
1	1000
...	...

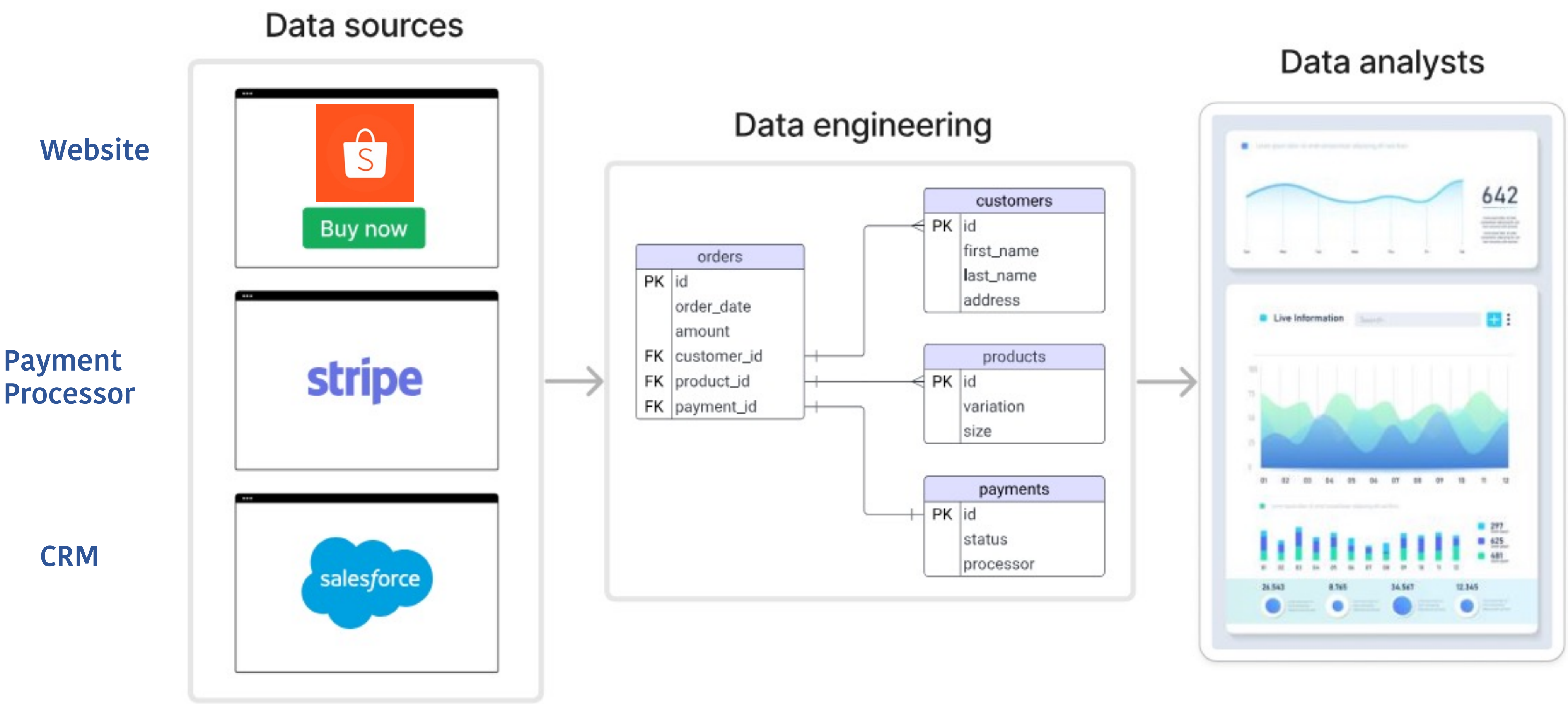
[Order]

[Finance Manager]

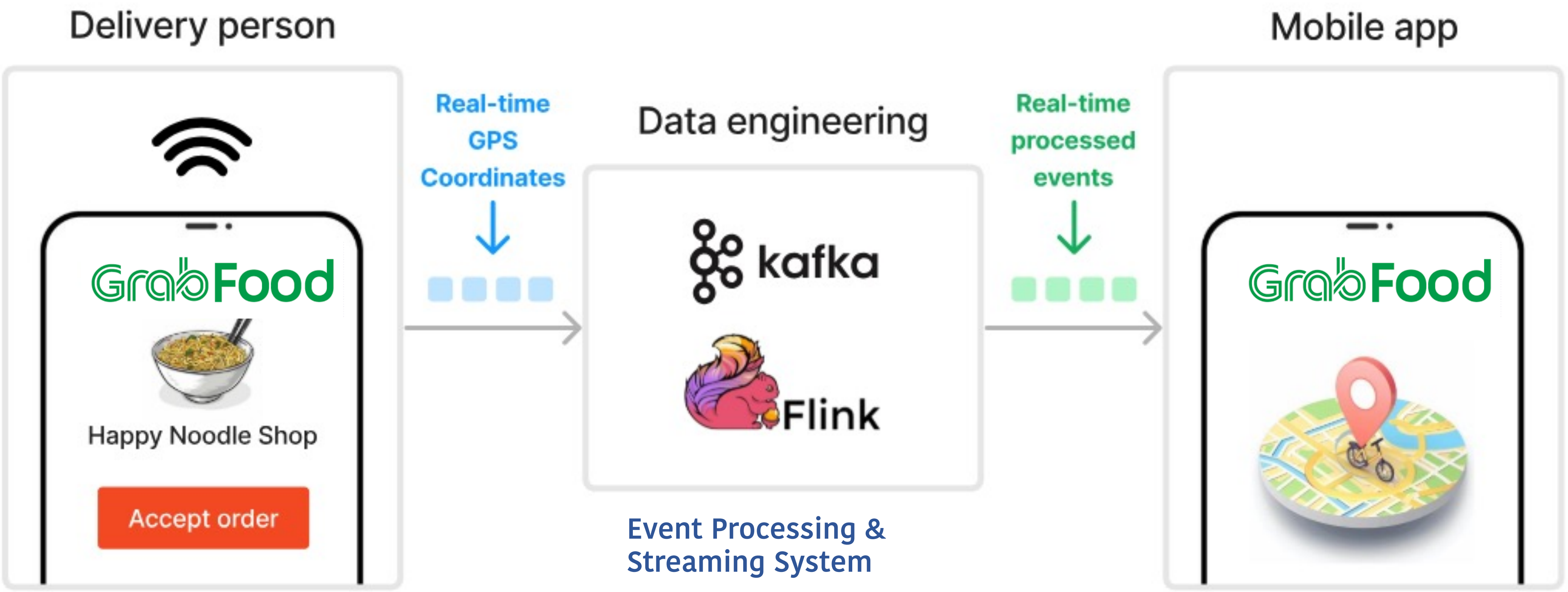
Many producers and consumers



Example - Business analytics (or BI)



Example - Real-time



Event Processing &
Streaming System

Why is it important?

Explosion of data sources

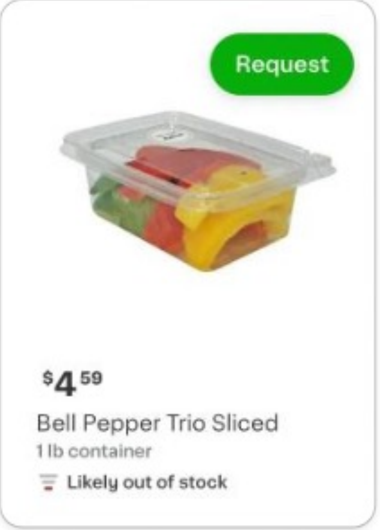
mailchimp Dropbox PostgreSQL mongoDB ups zendesk

Google Analytics stripe slack KLAVIYO shopify

Why is it important?

Explosion of data uses

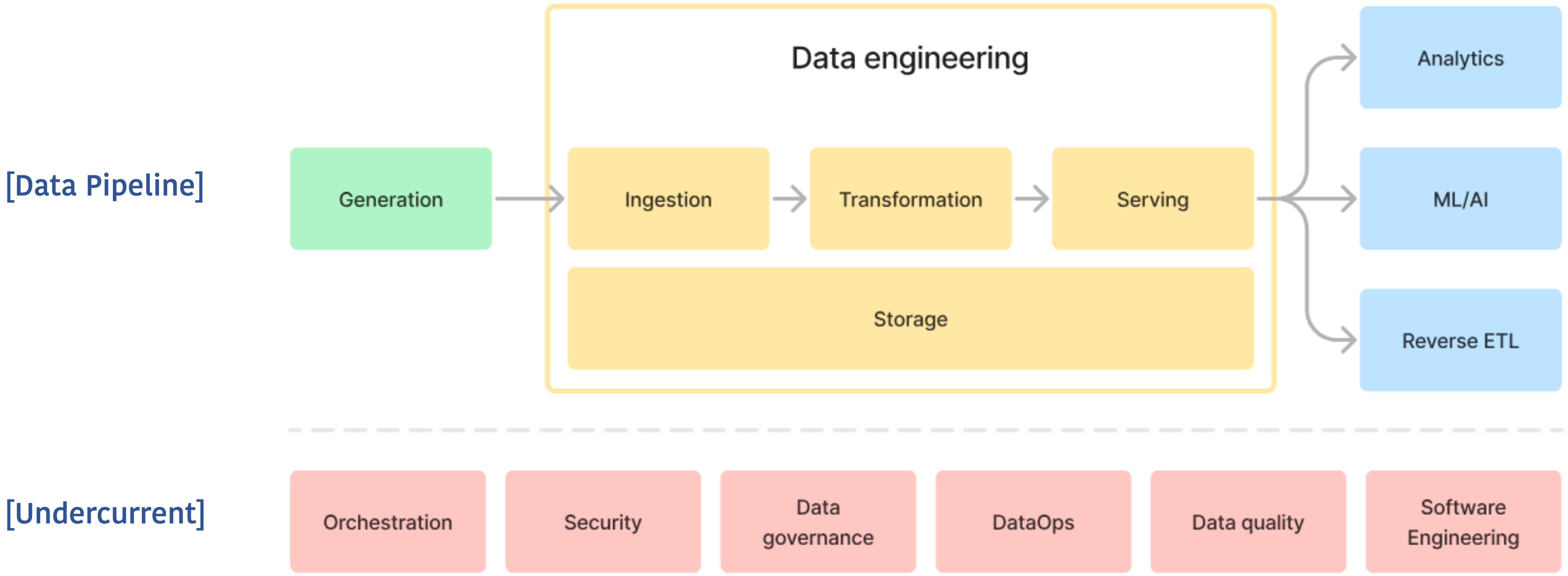
instacart



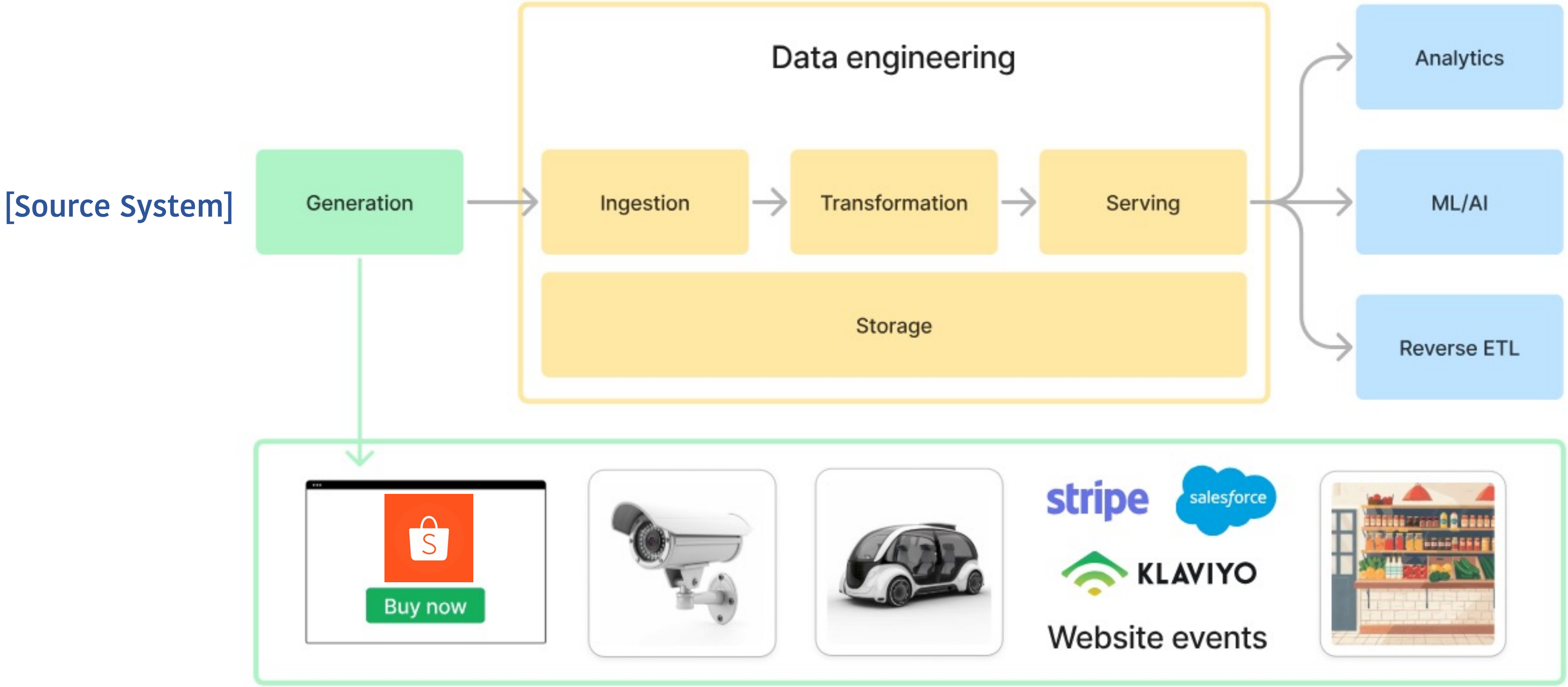
Demand Forecasting

AI

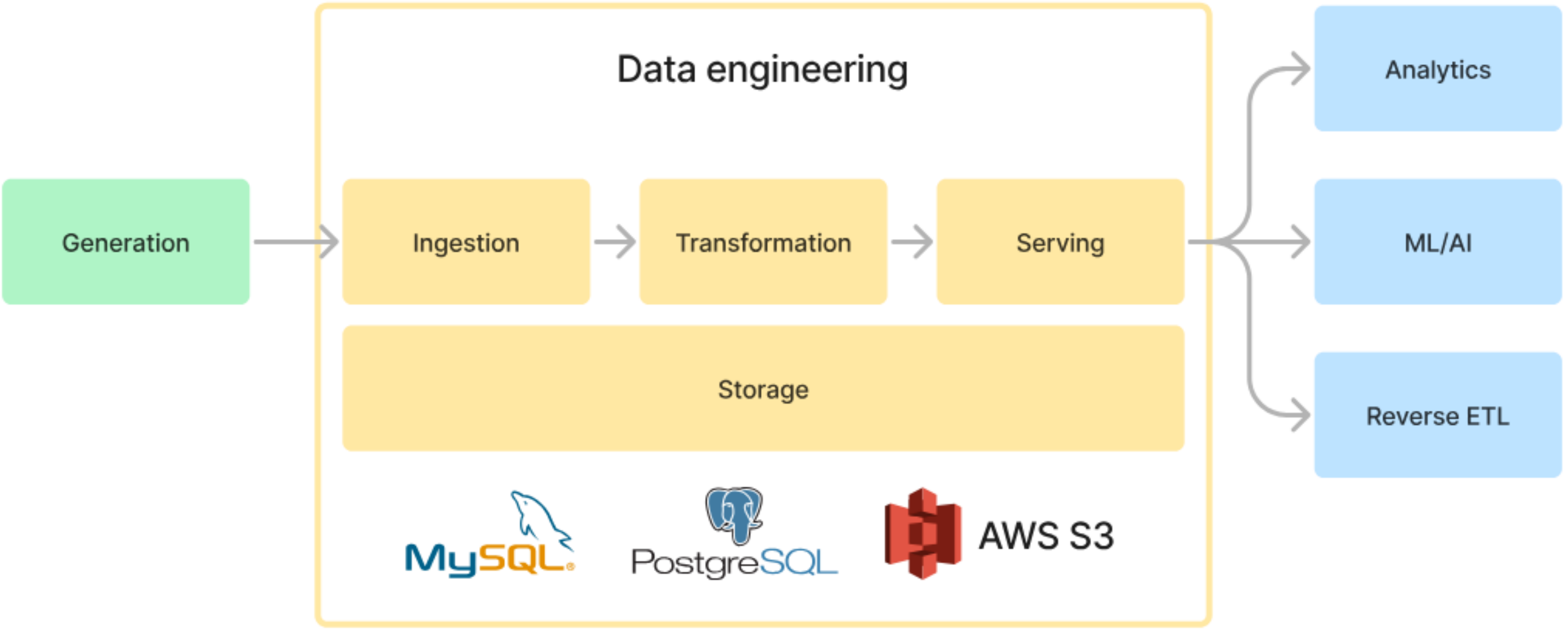
End-to-end data pipeline



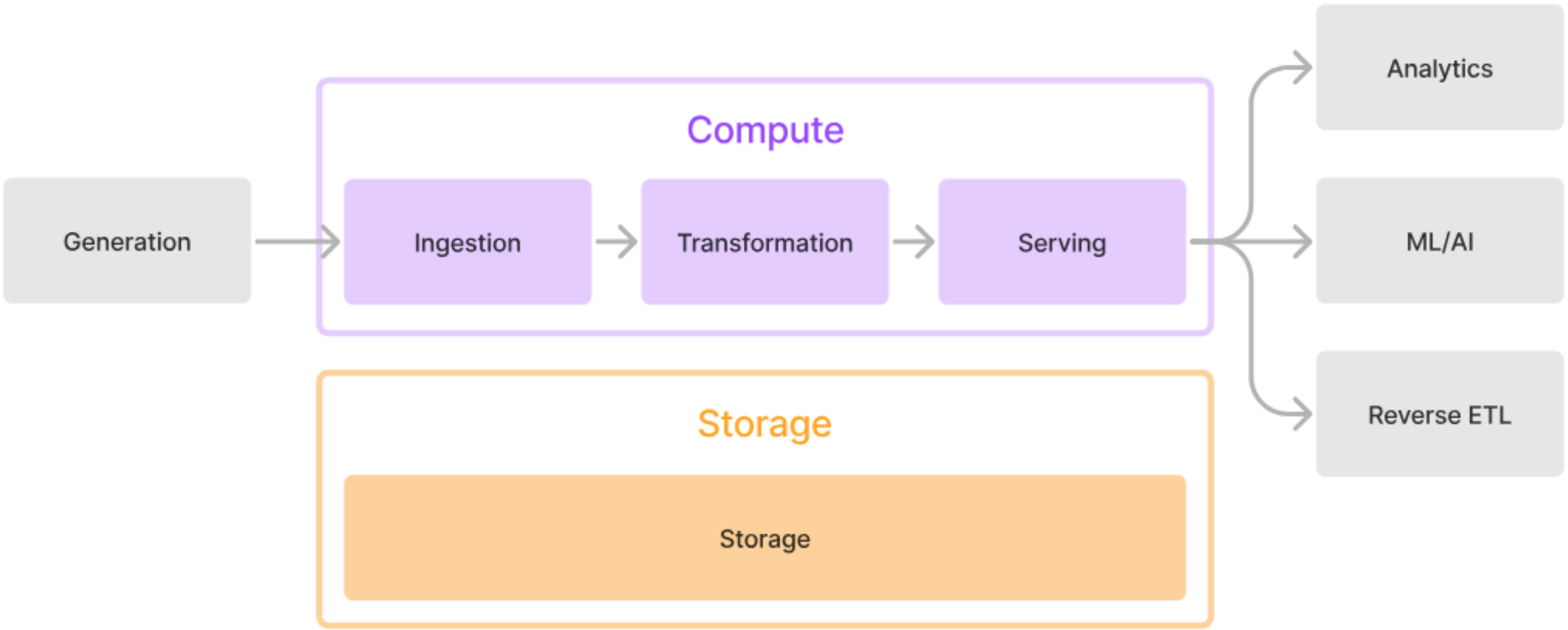
End-to-end data pipeline



End-to-end data pipeline



Compute vs Storage



Undercurrents

Orchestration	Security	Data governance	DataOps	Data quality	Software Engineering

1980-2000: Data warehouses

Data warehouses
+
Scalable analytics (BI)

MPP:
Massively Parallel Processing

BI engineer
ETL developer
Data warehouse engineer

On-prem (vs the Cloud)
Monolithic data stores

2000s: The beginning of “Big Data”

Internet + Google, Amazon, etc.

The birth of the Cloud
(built on cheap, commodity hardware)

AWS
starting with S3 and EC2

2003 Google publishes a paper on
MapReduce - Hadoop

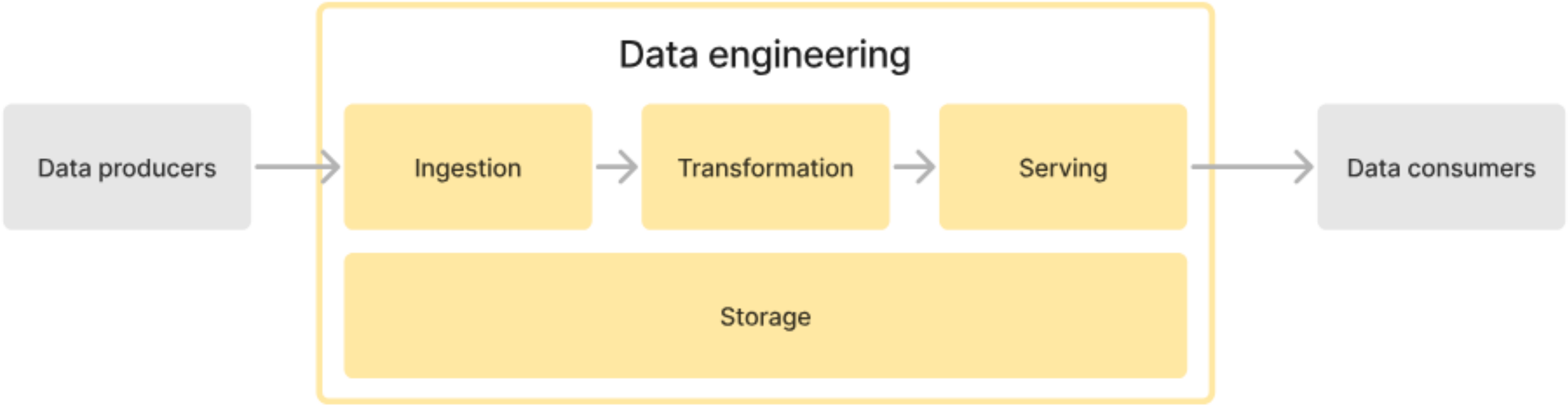
2000s-2010s: Big Data engineering

Big Data = Hadoop

Hadoop Ecosystem:
Hadoop, YARN, HDFS

- A lot of work to manage
- Requires a specialized engineering team

2020s: Modern Data Stack



[Compose together Lego-like modular components]

[No single tech stack that makes up the entire pipeline]

Data maturity : determines the complexity of your data pipeline.

Start → Scale → Lead

	Start	Scale	Lead
Typical use case	Data analytics	+ Data products	+ Real-time, ML/AI
Data volume	Small	Medium to large	Large to huge
Team size	Small team	Bigger team	+ Data science team
Tech maturity	Very basic (e.g. monolithic DB)	Proper data practices (e.g. automation, DevOps)	ML/AI features Enterprise features

Data engineer’s place within the data team

Responsibilities - business & technical

Business responsibilities

- Communicate with both technical and non-technical people
- Understand how to scope and manage a data project
- Minimize cost and work within budget

Technical responsibilities

- Create a good data architecture
- Build and manage a reliable and performant data pipelines
- Security, data governance, automation, observability, etc.

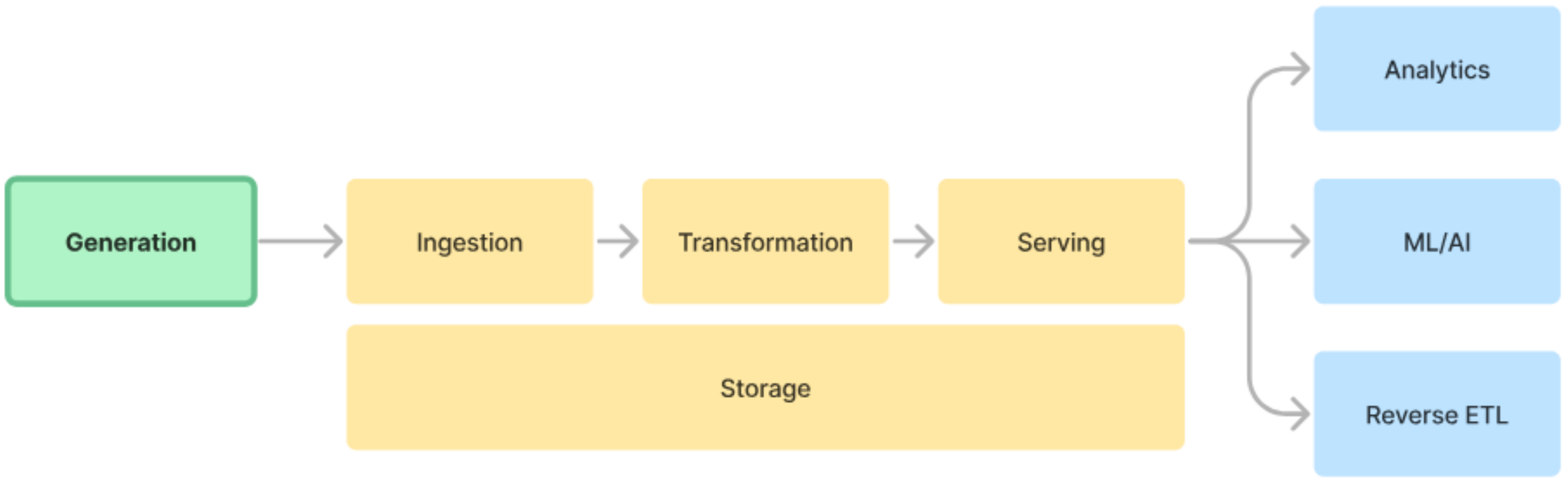
Data Engineers:

Required technical skills

SQL	Python	Basic DevOps (e.g. Docker)	Bash

1 Data Generation

Generation [Source Systems]



Structured vs unstructured data

Structured data

order_id	user_id	amount	...
1	3	$20	...
2	61	$25	...
3	33	$10	...

SQL

BI, Classical ML (small # of features)

Unstructured data

- Logs or events
- Images
- Audio
- Video
- Free-form text
- ...

Deep Learning

Neural networks (large # of features)

Types of databases

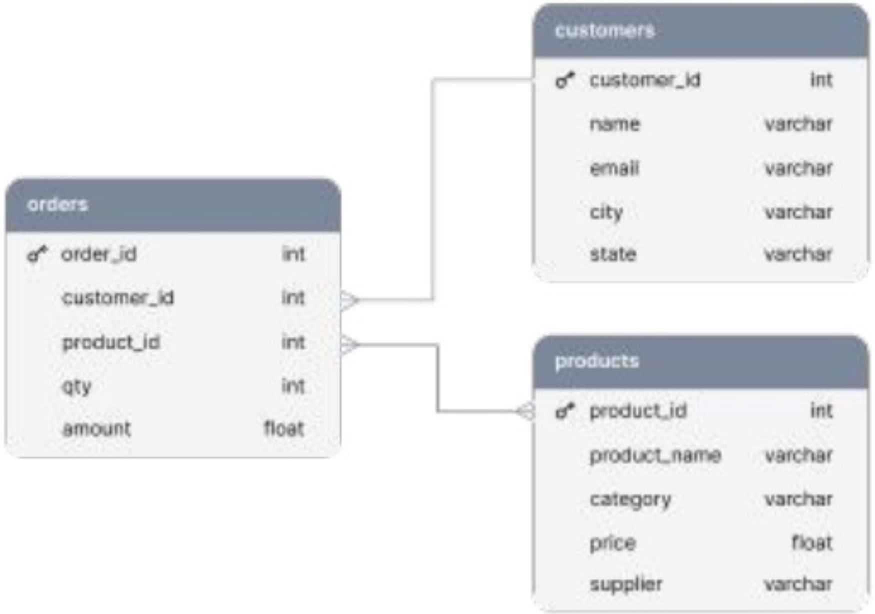
Relational database (RDBMS)

Non-relational database (NoSQL)

Key-value store

Relational database (RDBMS)

order_id	amount	...
1	$20	...
2	$25	...
3	$10	...



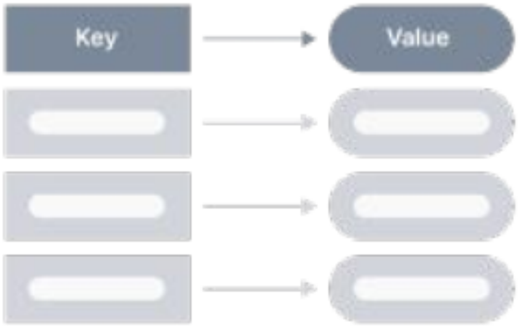
NoSQL database

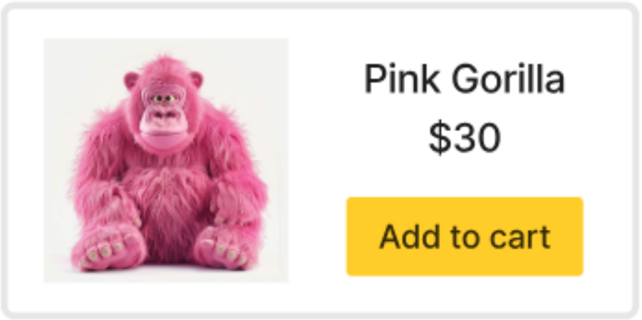
Relational databases	NoSQL databases
Tabular structure	JSON structure
Strict schema	Flexible schema
Normalized	De-normalized
Single-machine	Distributed

customer_id	name	...
1	John Doe	...
2	Jane Doe	...
3	Alan Smith	...

```json
{
  "customer_id": 1693,
  "name": "John Doe",
  "email": "john@email.com",
  "address": {
    "street": "123 Heaven St",
    "city": "Rochester",
    "state": "NY"
  },
  "phone": "+123 456 789"
}
```

Key-value store





Memory-based

Caching application data
Very fast lookup / High concurrency
Data not persisted

Persisted

Fast performance
Simple / Single table
Easy to use (store anything in value)
Highly scalable

Third-party systems via API

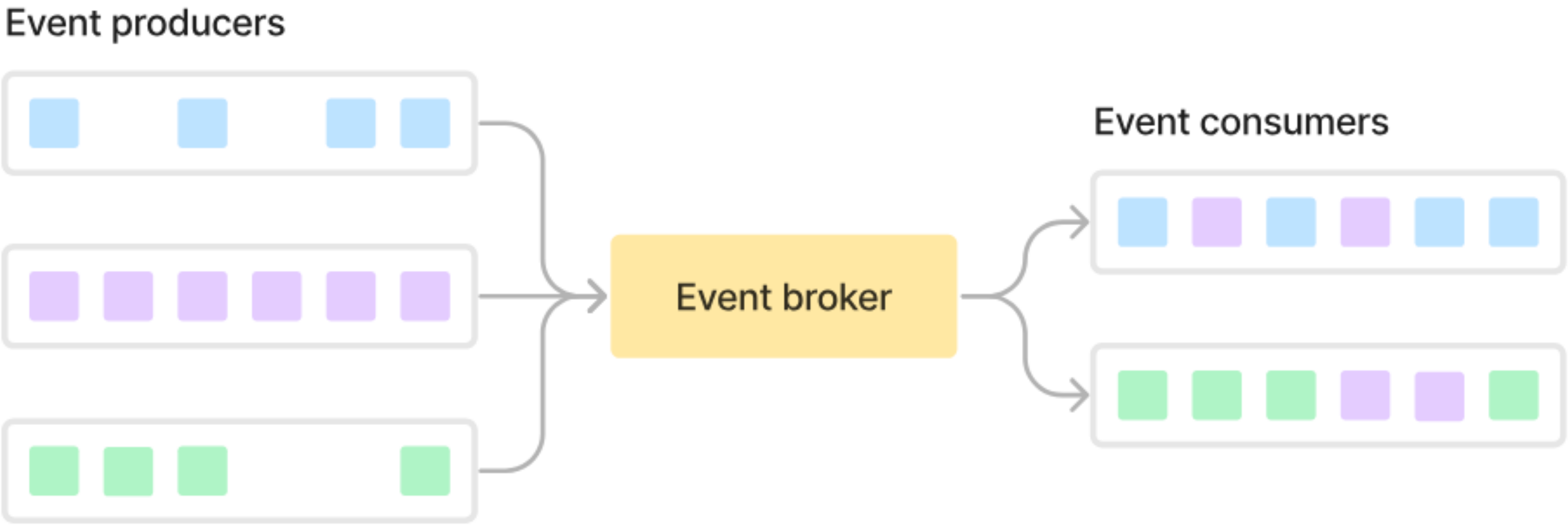
REST API

GraphQL

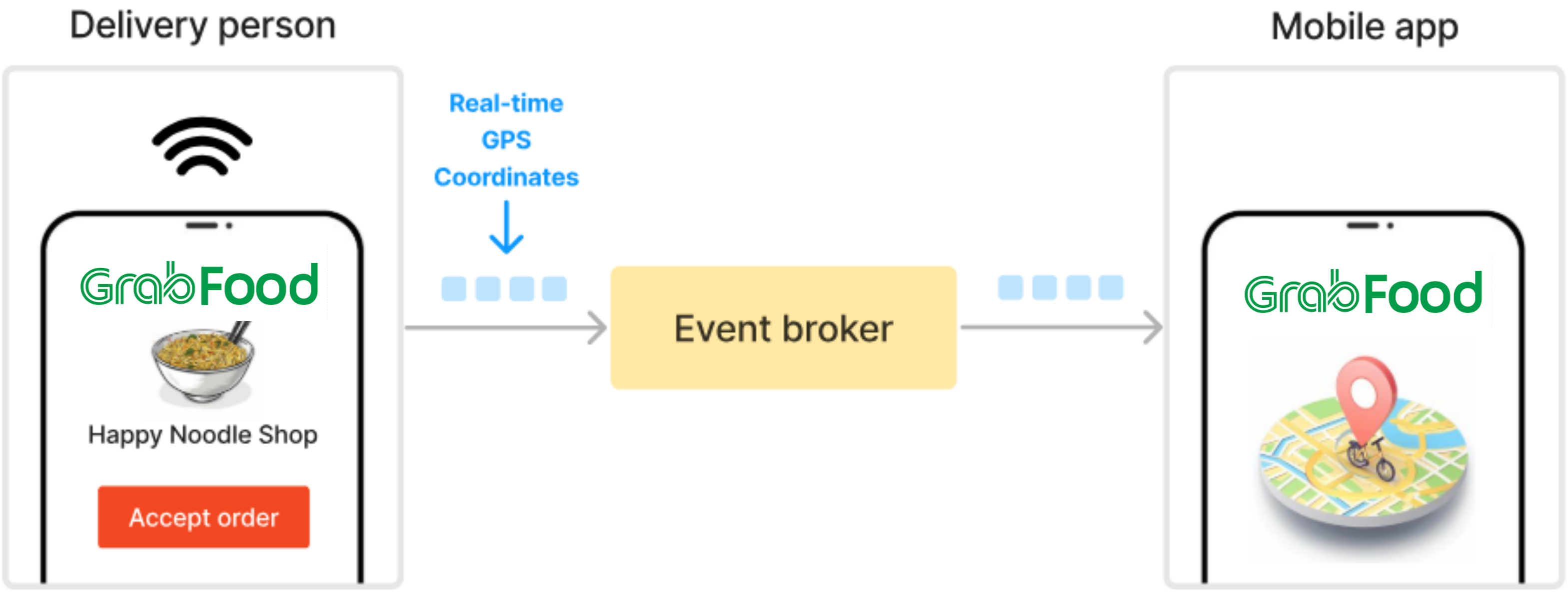
gRPC

Dropbox
salesforce
zendesk
Google Analytics
shopify
ups
KLAVIYO
stripe
Etsy
mailchimp
slack
HubSpot

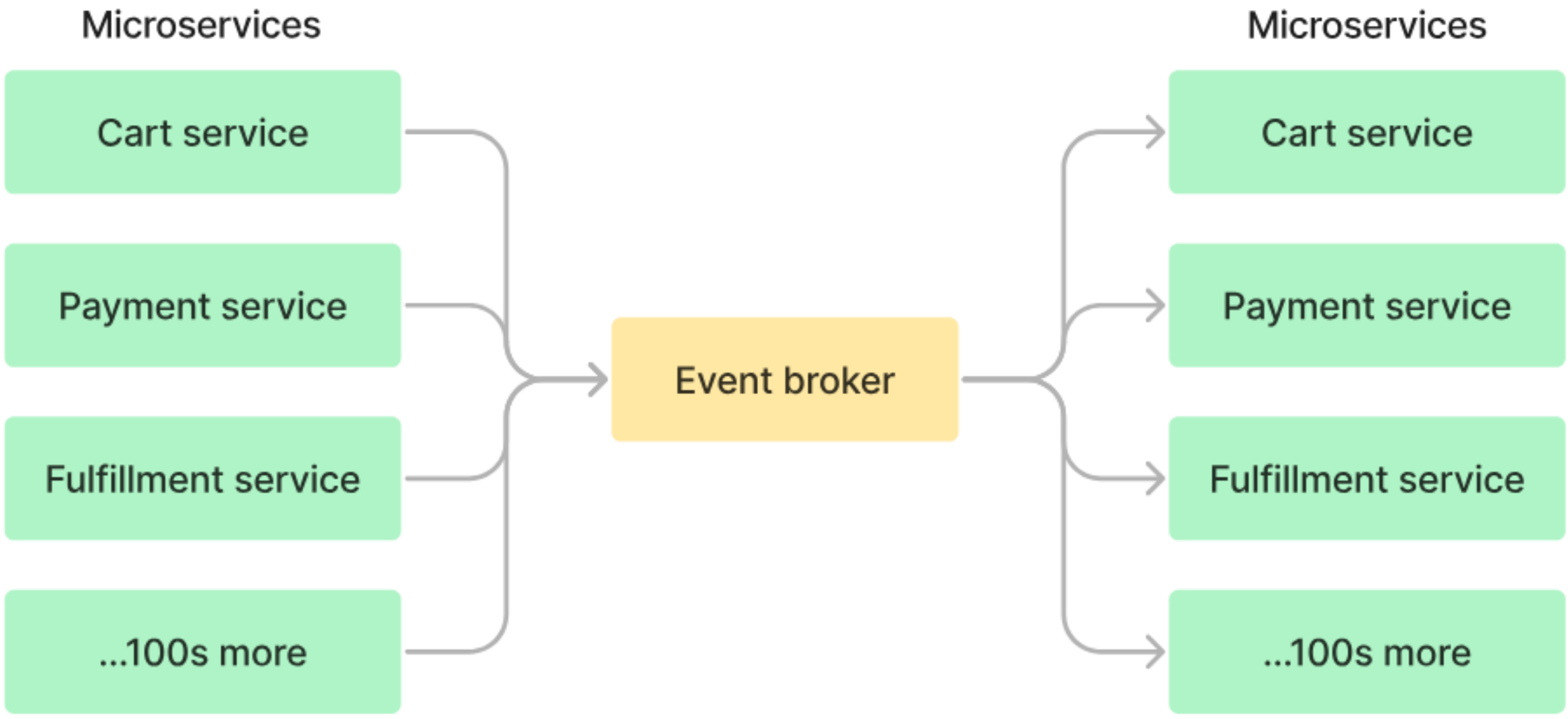
Event streams



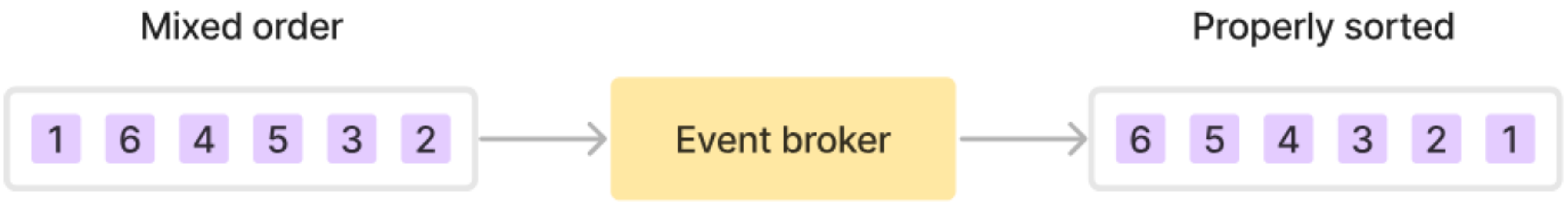
Event streams



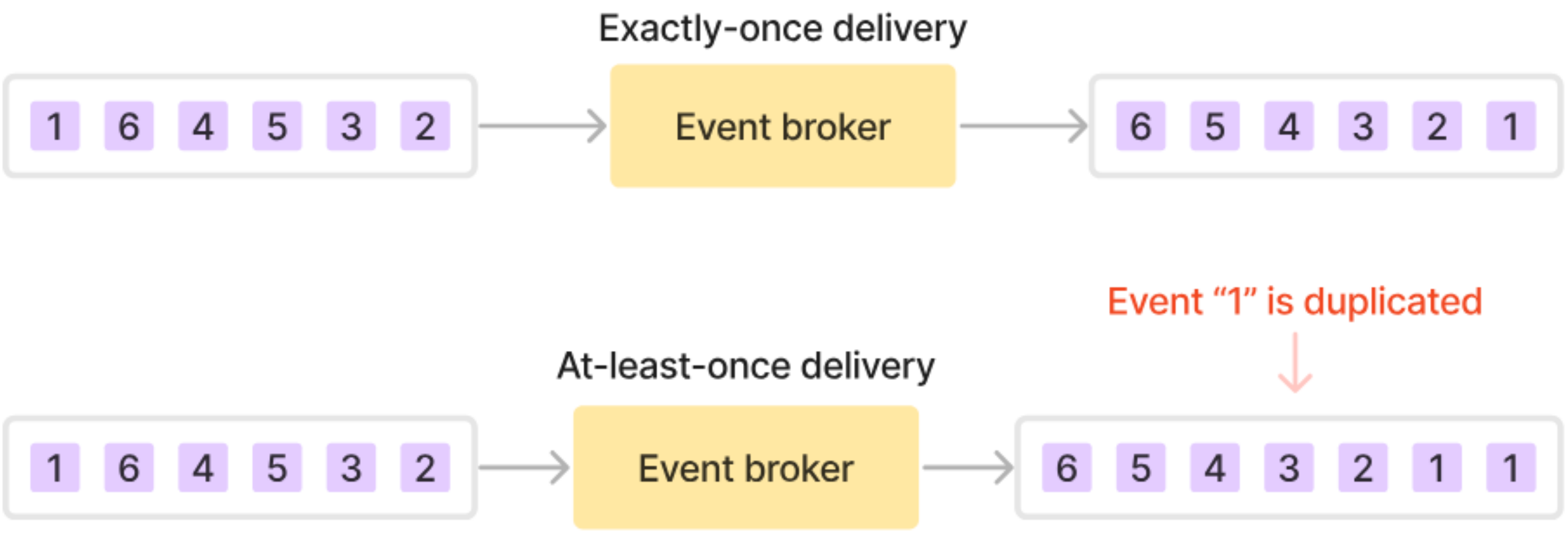
Event streams



Messages arriving in the wrong order



Exactly-once vs at-least-once delivery



Idempotency

[An operations is “idempotent” when the same result comes out no matter how many times you run it.]

[Insert : Not Idempotent]

INSERT operation

id	row_id	...
1		
...		
100		

INSERT row_id = 123

id	row_id	...
1		
...		
100		
101	123	

INSERT row_id = 123

id	row_id	...
1		
...		
100		
101	123	
102	123	

[Merge : Idempotent]

MERGE operation

id	row_id	...
1		
...		
100		

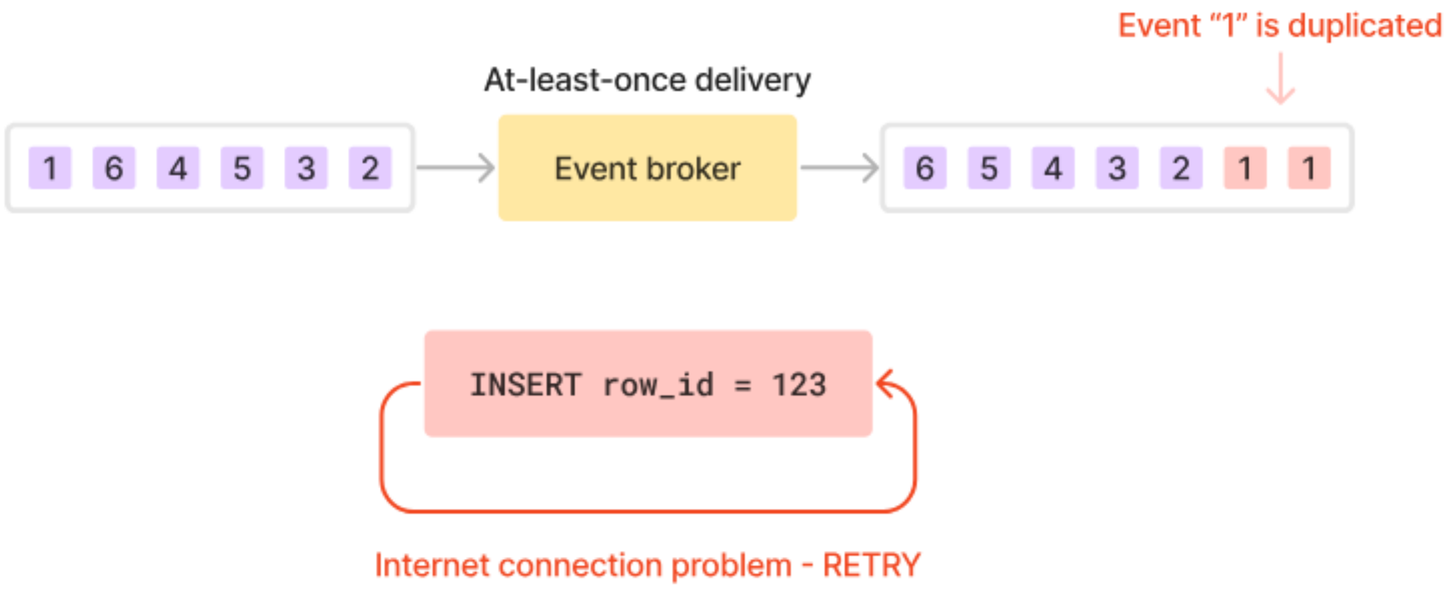
MERGE row_id = 123

id	row_id	...
1		
...		
100		
101	123	

MERGE row_id = 123

id	row_id	...
1		
...		
100		
101	123	

Why is idempotency so important?

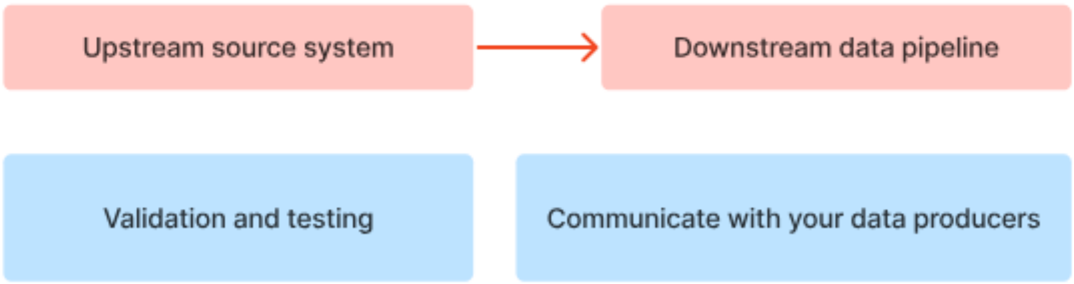


Popular event streaming platforms

AWS SQS	Amazon Kinesis	RabbitMQ	Kafka	Pulsar	Spark

Caveats

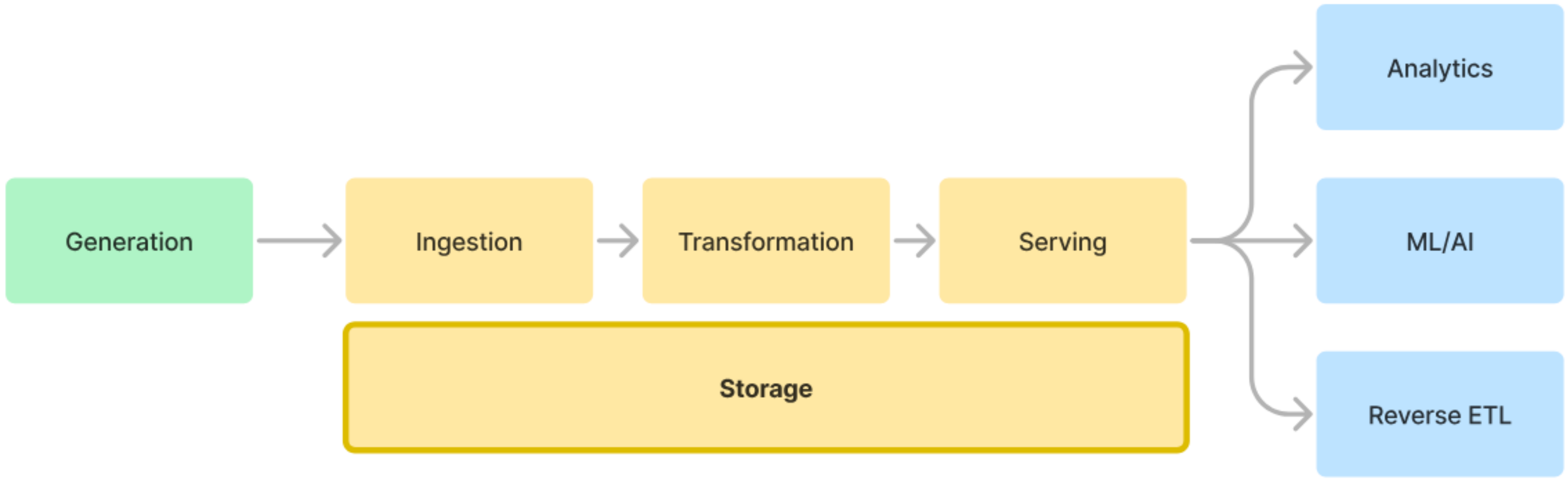
Do NOT consider the source systems as someone else's problem



2

Data Storage

Storage


Generation
Ingestion
Transformation
Serving
Storage
Analytics
ML/AI
Reverse ETL

Storage hardware

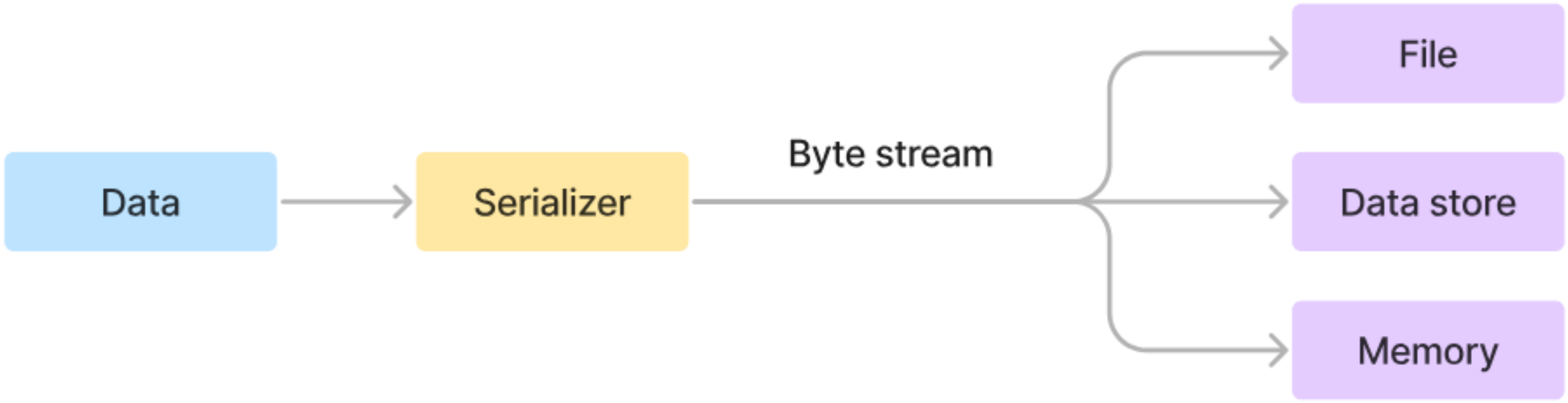
HDD (Hard Disk Drive)

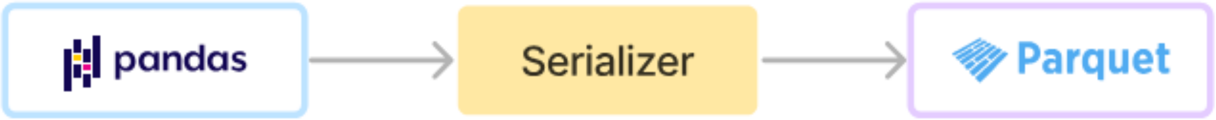
SSD (Solid State Drive)

Memory (RAM or Random Access Memory)

Storage formats - serialization

: Turning data into byte streams to easily save or transport it





We serialize the data into standard format which is sent around and deserialized on the receiving end

Storage formats - serialization

Row-based serialization file formats

- XML
- JSON
- CSV

- Fast lookup of individual rows

Column-based serialization file formats

- Parquet
- ORC
- Avro
- Arrow

- Aggregation by Column

Row-based databases

Columnar databases

Compression and caching

Compression

- Reduce the amount of data stored
- Faster query (less to search)
- Faster transport (less data to move)

Caching

- Some storages are faster to access
- Memory is 1,000x faster than SSDs and CPU caches are orders of magnitude faster than memory
- Faster storage is more limited and expensive

Single machine vs distributed storage

Vertical scaling

5 vCPU
10TB

→

20 vCPU
40TB

Horizontal scaling

5 vCPU
10TB

→

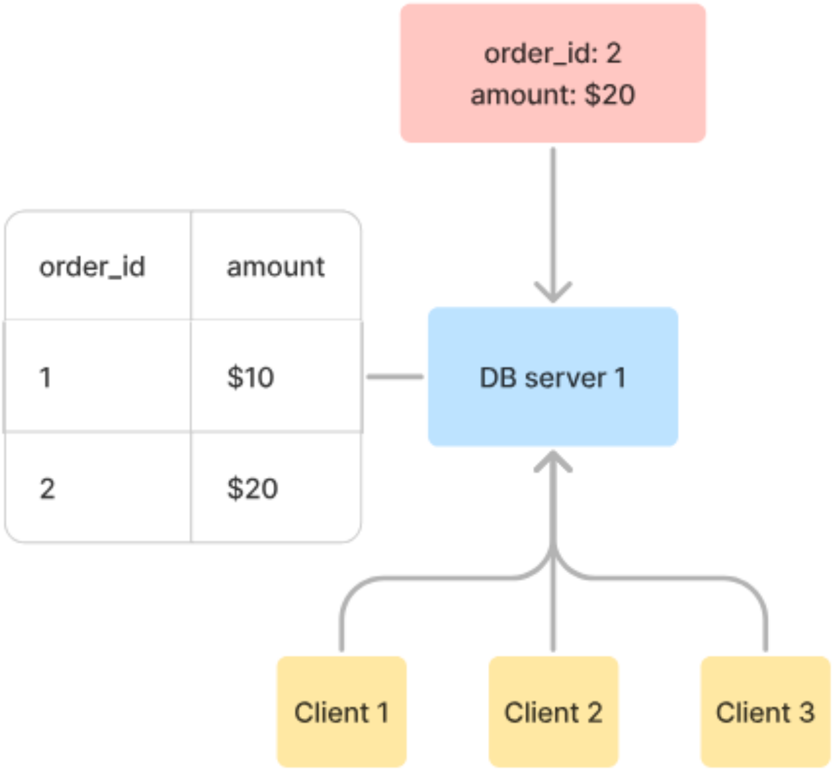
5 vCPU 10TB	5 vCPU 10TB
5 vCPU 10TB	5 vCPU 10TB

Distributed Storage + Distributed Compute

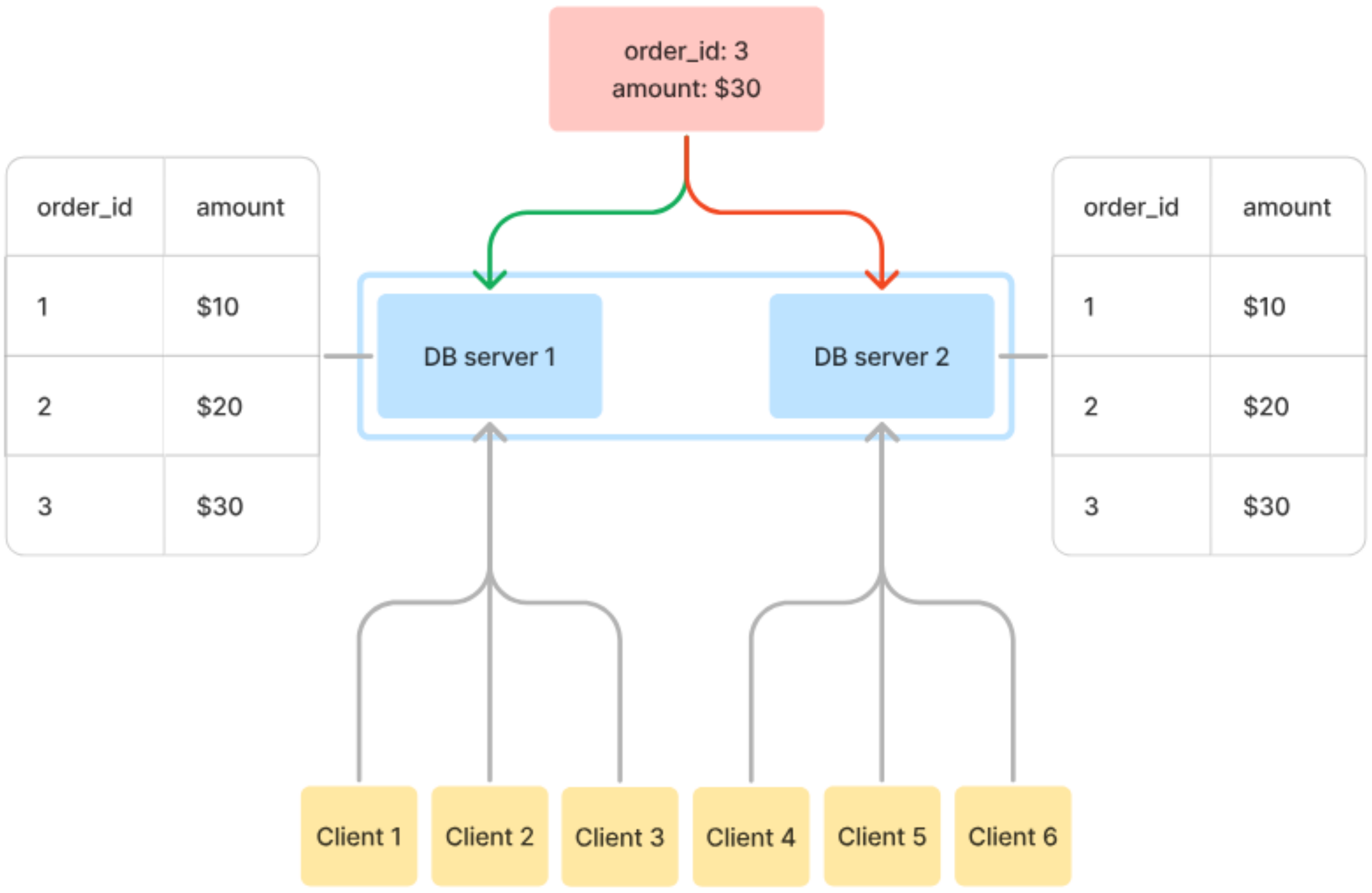
Spark trino amazon REDSHIFT snowflake

Strong vs eventual consistency

Strong consistency



Eventual consistency



ACID vs BASE

Single-machine transactional database

ACID

- Atomicity
- Consistency
- Isolation
- Durability

Distributed database

BASE

- Basically Available
- Soft-state
- Eventual consistency

Type of storage systems

File storage

- Local fs
- NAS
- Cloud fs

Block storage

- HDD/SSD
- AWS EBS

Object storage

- AWS S3

Cache storage

- RAM
- Memcached
- Redis

Streaming storage

- Buffering
- Often ephemeral, but can be persisted

File storage vs object storage

Scale	Structure	Latency	Updates
File <= Millions	File = Tree	File = Fast	File = Mutable
Object = ∞	Object = Flat	Object = Slow	Object = Immutable

OLTP vs OLAP (or Row vs columnar)

order_id	customer_id	order_date	amount
1	a	Jan 29	1000
2	f	Jan 30	1250
...	...	...	...
100	d	Mar 10	3500

OLTP (row-based database) in memory:

✓ Specific row, Single transactions
✓ Online Transaction Processing (OLTP) : row-based database (e.g. PostgreSQL, MongoDB)

1	a	Jan 29	1000	2	f	Jan 30	1250	...	100	d	Mar 10	3500

OLAP (column-based database) in memory:

✓ Analytics use case (aggregating total Orders, total customers etc.)
✓ Online Analytical Processing (OLAP) : column-based database

1	2	...	100	a	f	...	d	Jan 29	Jan 30	...	Mar 10	1000	1250	...	3500

OLTP vs OLAP (or Row vs columnar)

HDD (Hard Disk Drive)

OLAP (columnar) databases

Volumes are important

SSD (Solid State Drive)

OLTP (row-based) databases

Speed is important

Data warehouse, lake, lakehouse

Data warehouse

amazon REDSHIFT · Google Big Query · snowflake

- OLTP → OLAP
- Well organized

Data lake

AWS S3

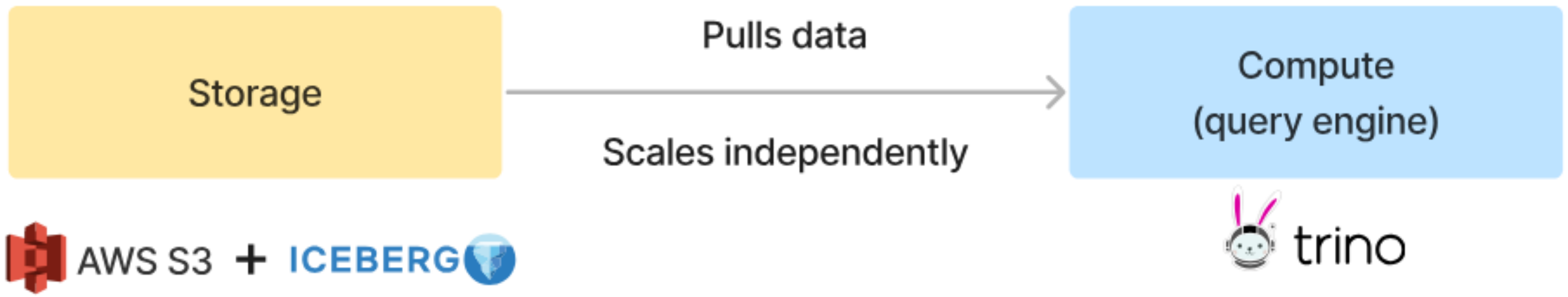
- Inexpensive object storage
- Structured & unstructured
- Easily becomes "data swamps"

Data lakehouse → Object storage (AWS S3) + Open table format (ICEBERG)

ICEBERG · Apache hudi · DELTA LAKE

- OLAP
- Advantage of the warehouse + lake
- Good integration with external query engines

Separation of compute from storage



Data storage lifecycle - hot, warm, cold

High data access frequency → Low data access frequency

Hot	Warm	Cold
AWS S3	AWS S3 Infrequence access tier	AWS Glacier
Azure Data Lake Storage	Azure Data Lake Storage	Azure Data Lake Storage

Hot
- Accessed often
- Expensive
- Fast

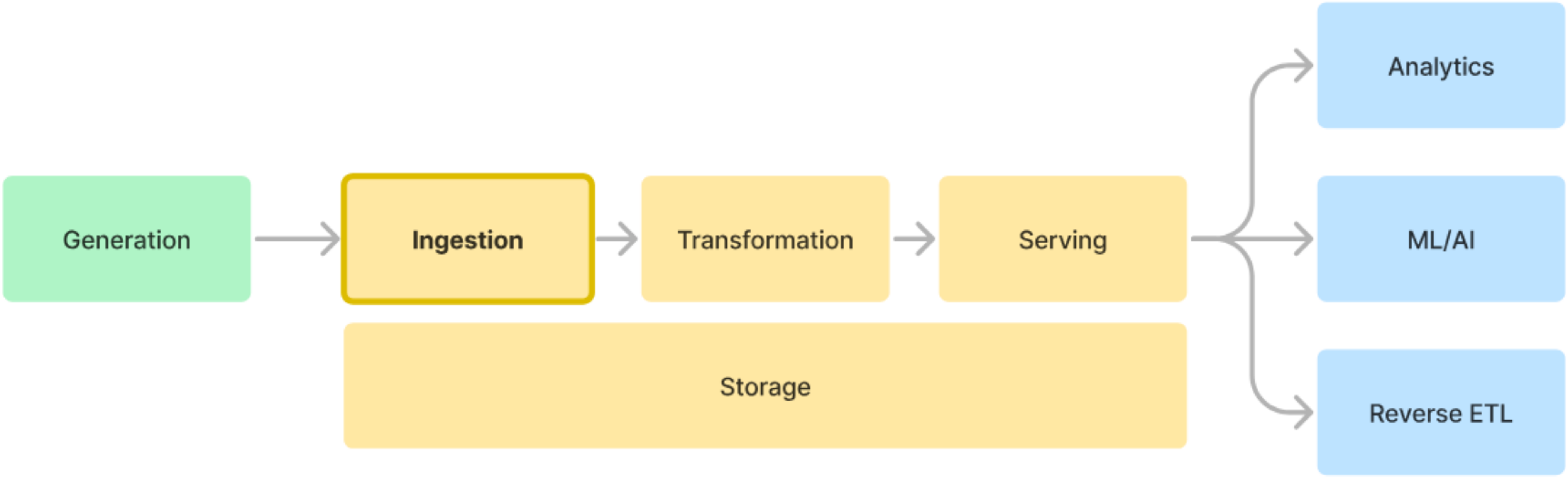
Warm
- Accessed infrequently
- Inexpensive

Cold
- Rarely accessed
- Very cheap

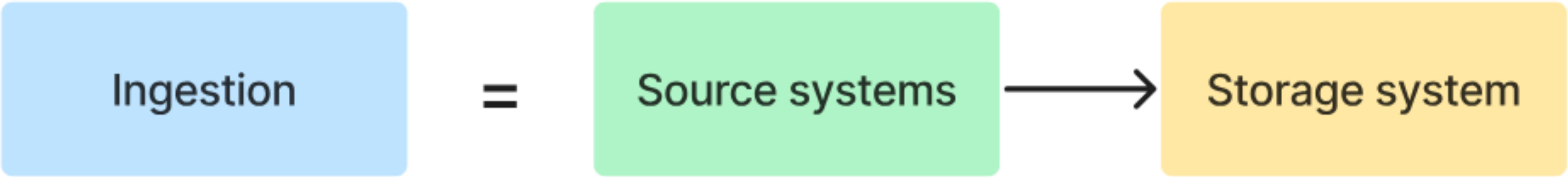
3.1

Data Ingestion

Ingestion


Generation
Ingestion
Transformation
Serving
Storage
Analytics
ML/AI
Reverse ETL

Ingestion



Frequency of ingestion: batch, micro-batch, streaming

Batch

- Daily or hourly
- Ex. Tax reporting

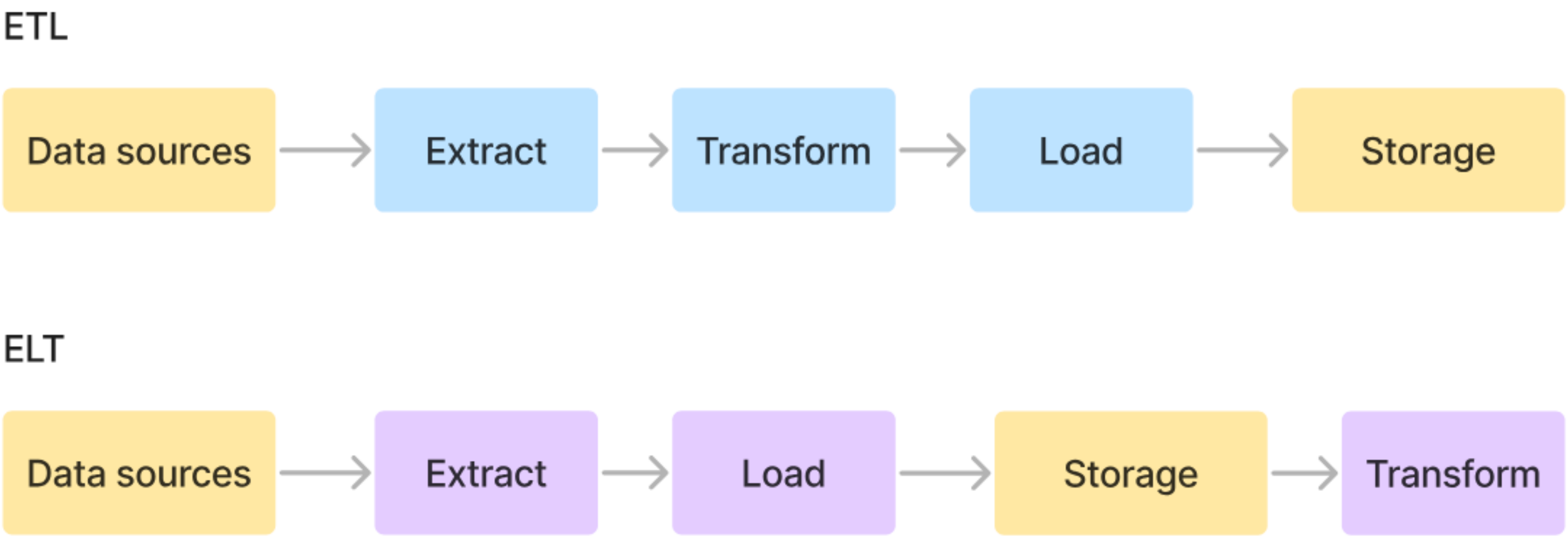
Micro-batch

- Minutes to seconds
- Ex. Near real-time analytics

Streaming

- Seconds to sub-seconds
- Ex. Uber

ETL vs ELT


ETL
Data sources
Extract
Transform
Load
Storage
ELT
Data sources
Extract
Load
Storage
Transform

Scalability and schema changes

Scalability

- Make sure your system can scale up and down automatically based on workload
- For streaming data, add a sufficient buffer for fluctuating event volumes

Schema changes

- Upstream changes to fixed schema will break the ingestion pipeline
- Schema migration features help, but not perfect - frequently communicate with upstream teams

Ways to ingest data

DB connection JDBC

CDC PostgreSQL MySQL

Ways to ingest data - CDC

Customers table

customer_id	Address
1	123 Disney St
...	...

COPY (first time)

Data Lakehouse

customer_id	Address
1	123 Disney St
2	7 Another St
...	...

COPY (second time)

Data Lakehouse

Customers table

customer_id	Address
1	123 Disney St
...	...

customer_id	Address
1	456 Pixar Way
...	...

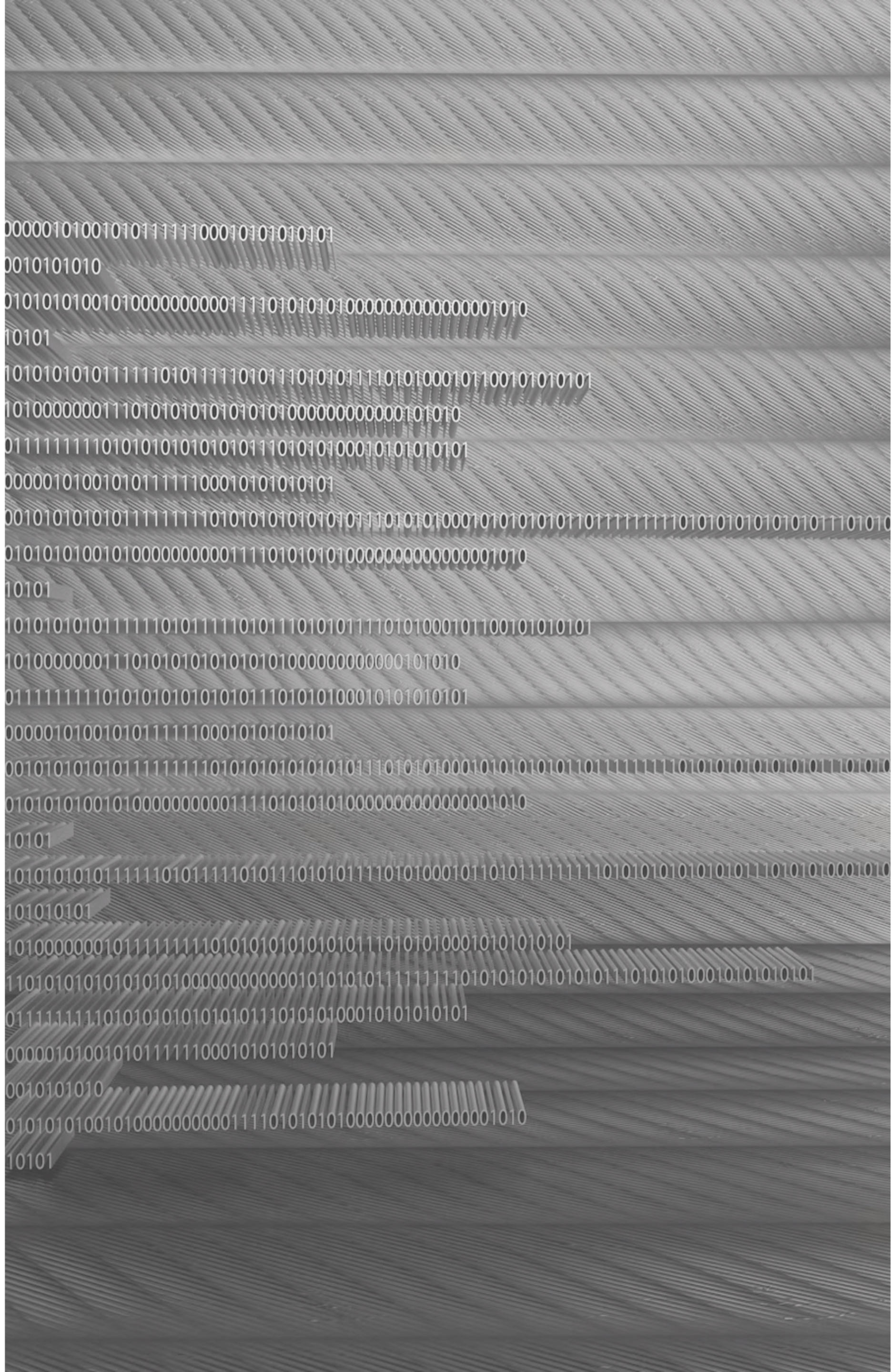
Customers table log

```
{
   op: “update”,
   field: “address”,
   before: “123 Disney St”,
   after: “456 Pixar Way”
}
```

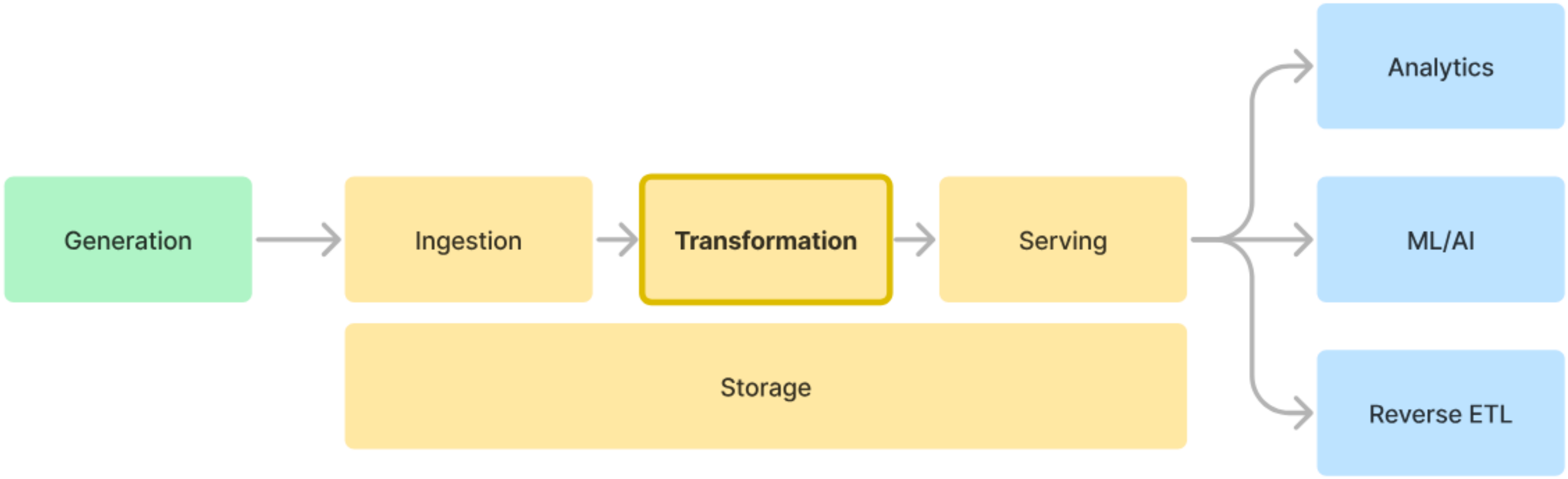
Ways to ingest data

APIs	salesforce, Google Analytics, KLAVIYO
Managed connectors	Fivetran, Airbyte, meltano from arch
Streaming	Amazon Kinesis, kafka, PULSAR

3.2 Data Transformation



Transformations



Queries

order_id	amount	...
1	1250	...
...	...	...
98292	2000	...

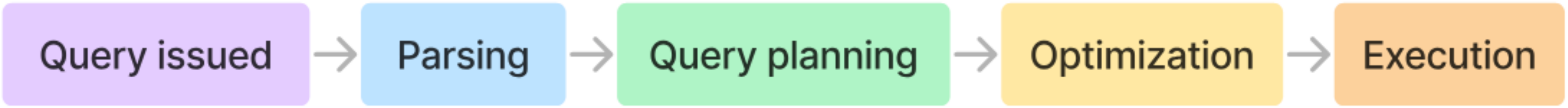
```
SELECT * FROM orders WHERE order_id=1

ALTER TABLE orders ADD shipping_amount INT
```

Queries - DDL, DML, DCL

DDL (Data definition language)	`CREATE table customers`
DML (Data manipulation language)	`SELECT * FROM orders WHERE order_id=1`
DCL (Data control language)	`GRANT SELECT ON main_db TO user_name Dave`

What happens when you run a query?



Improving the query performance

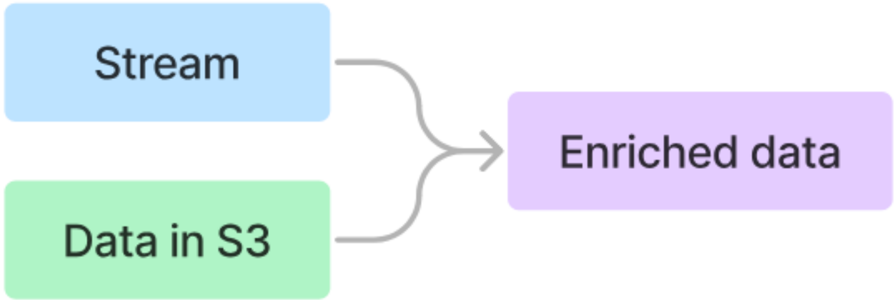
- Reduce the search space (prune, filter, etc)
- Use “EXPLAIN” SQL statement
- Pre-join frequently joined tables
- Vacuum dead records
- Leverage caching

Querying streaming data

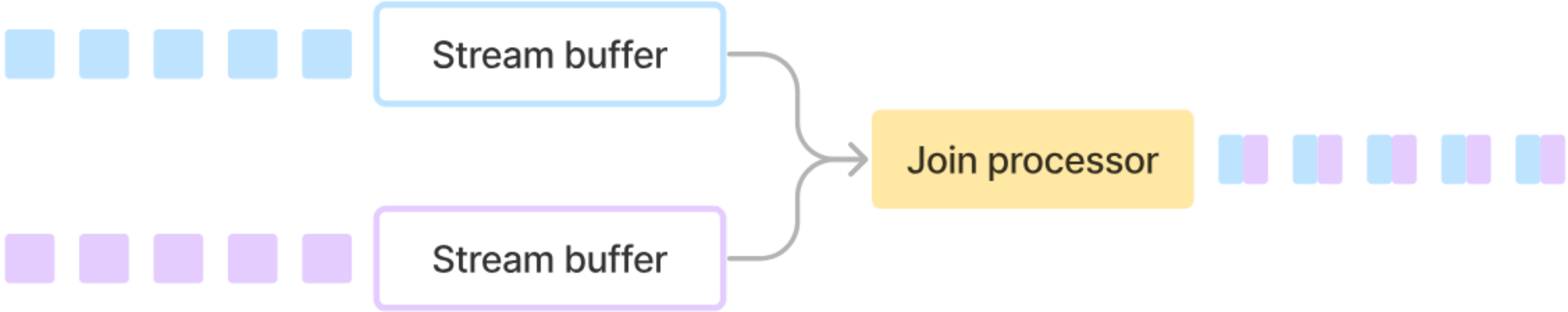
Windowing

- Session Window
- Fixed Time Window
- Sliding Window

Enrichment (stream + other data)



Stream-to-stream joining

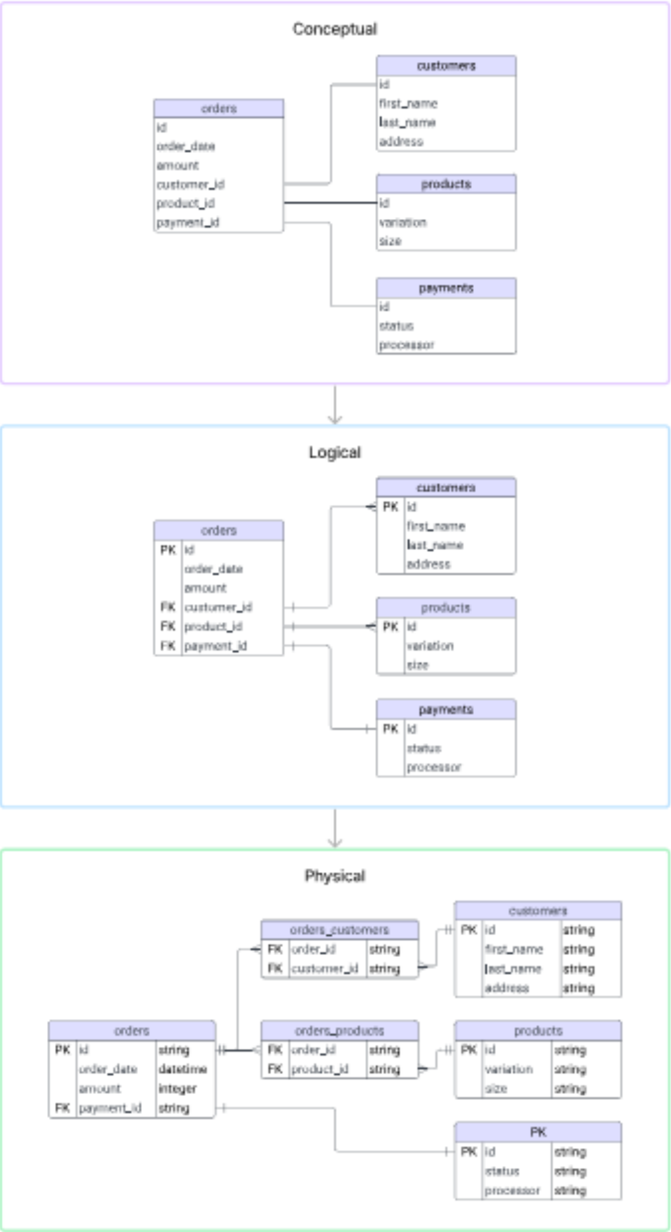


Data modeling

A data model is a simplified version of real-world business activities

Conceptual → Logical → Physical

Data modeling



Why do we need data modeling?

Easier to understand how different business data are connected

Allows you to have consistent definitions

Ingredients of successful data modeling

Communicate with the data consumers from the beginning

Model the data at the lowest granularity possible

Normalization

Data modeling with no duplicates

DRY Principle - “Don’t Repeat Yourself”

Normalization example

Denormalized “orders” table

id	order_date	order_items	customer_email	customer_name	amount	weight
1	Feb 12, 2024	MacBook, iPhone	seungchan@test.com	Seungchan Lee	350000	6
2	Feb 17, 2024	iPhone, iPad	nami@xyz.com	Nami Kim	250000	2.5
3	Mar 4, 2024	MacBook	nami@xyz.com	Nami Kim	250000	5
4	Apr 20, 2024	Vision Pro	nami@test.com	Nami Kim	300000	2

“orders” table

id	order_date	order_items	customer_id	amount
1	Feb 12, 2024	1, 3	1	350000
2	Feb 17, 2024	3, 4	2	250000
3	Mar 4, 2024	1	2	250000
4	Apr 20, 2024	2	2	300000

Normalized tables example

"orders" table

id	order_date	order_items	customer_id	amount
1	Feb 12, 2024	1, 3	1	350000
2	Feb 17, 2024	3, 4	2	250000
3	Mar 4, 2024	1	2	250000
4	Apr 20, 2024	2	2	300000

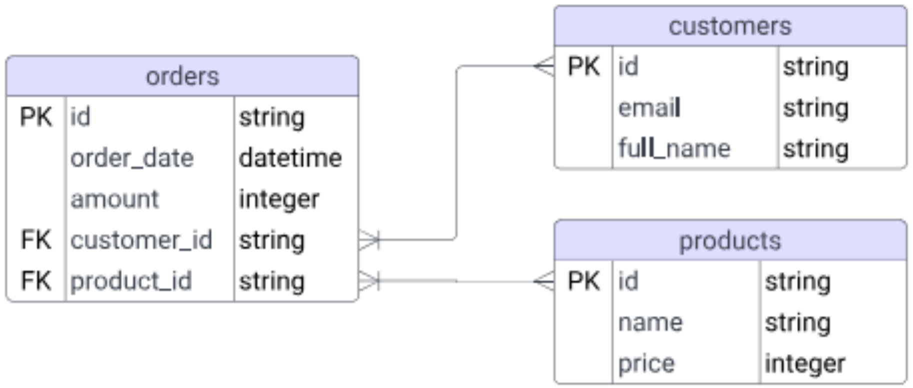
"products" table

id	name	price	weight
1	MacBook	250000	5
2	Vision Pro	300000	2
3	iPhone	100000	1
4	iPad	150000	1.5

"customers" table

id	email	full_name
1	seungchan@test.com	Seungchan Lee
2	nami@xyz.com	Nami Kim

Normalized tables - ER diagram



Different modeling techniques

Inmon

Highly Normalized Data

Kimball

Flexible and faster iteration
- Star Schema

Data vault

Hubs: Core business concept
Links: relationship between hubs
Satellite : store information about hubs

Wide tables

Join many related tables into a single highly denormalized table

Wide tables

“orders” table

id	order_date	order_items	customer_id	amount
1	Feb 12, 2024	1, 3	1	350000
2	Feb 17, 2024	3, 4	2	250000
3	Mar 4, 2024	1	2	250000
4	Apr 20, 2024	2	2	300000

“products” table

id	name	price	weight
1	MacBook	250000	5
2	Vision Pro	300000	2
3	iPhone	100000	1
4	iPad	150000	1.5

“customers” table

id	email	full_name
1	seungchan@deepintuitions.com	Seungchan Lee
2	nami@deepintuitions.com	Nami Kim

Wide tables - denormalized order items

id	order_id	order_date	order_item_id	order_item	customer_email	customer_name	amount	weight
1	1	Feb 12, 2024	1	MacBook	seungchan@test.com	Seungchan Lee	250000	5
2	1	Feb 12, 2024	3	iPhone	seungchan@test.com	Seungchan Lee	100000	1
3	2	Feb 17, 2024	3	iPhone	nami@test.com	Nami Kim	100000	1
4	2	Feb 17, 2024	4	iPad	nami@test.com	Nami Kim	150000	1.5
5	3	Mar 4, 2024	1	MacBook	nami@test.com	Nami Kim	250000	5
6	4	Apr 20, 2024	2	Vision Pro	nami@test.com	Nami Kim	300000	2

- Faster Analytical Queries

Transformations

```
select id as order_id from orders
```

↓

```
select count(order_id) as number_of_orders from orders
```

↓

```
left join customer_orders on customers.customer_id = customer_orders.customer_id
```

dbt Core

Source-conformed Data → Business-conformed Data

dagster

Orchestration

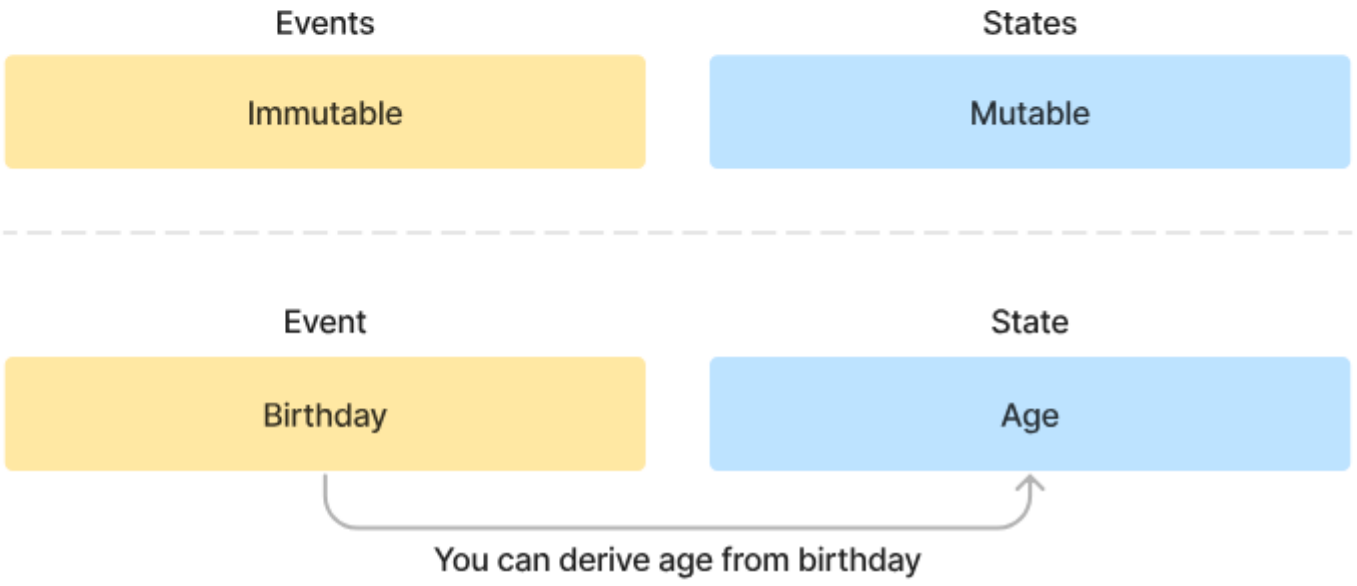
SQL vs Code?

Views and materializations

Tables	Views (~virtual tables)
Stored in physical storage	Not stored in physical storage
Retrieved from storage	Re-computed every time
Requires less compute	Requires more compute
Data could be stale	Data is always fresh

Materialization is a process of persisting the query in physical storage

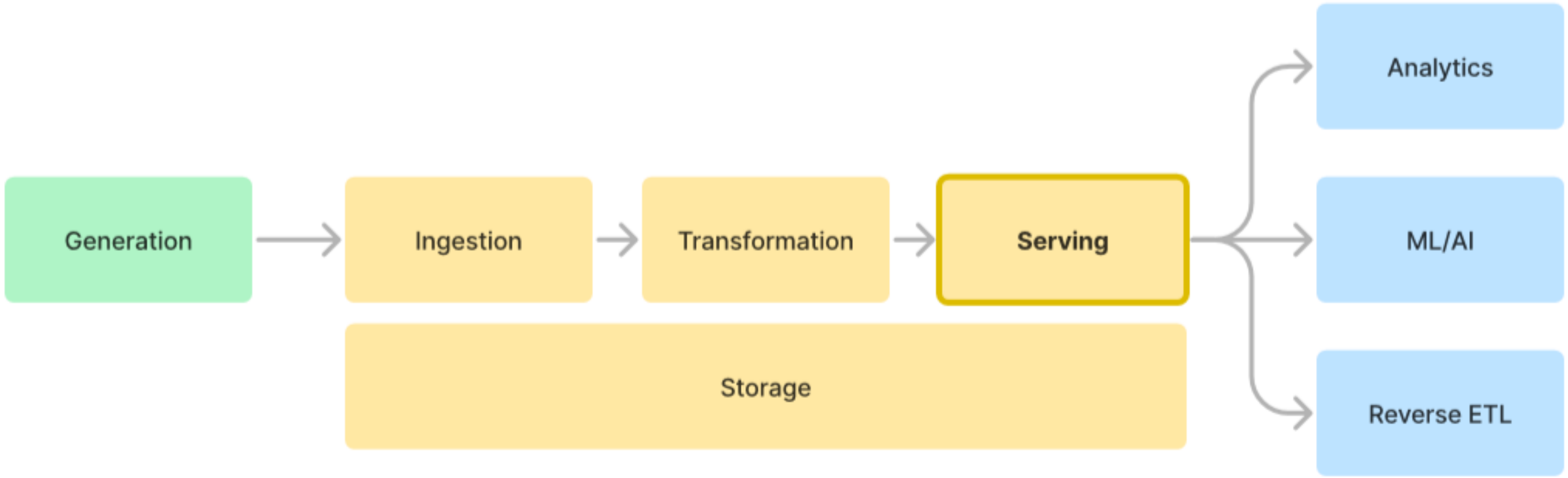
Events vs states



Streaming vs batch

3.3 Data Serving

Serving


Generation
Ingestion
Transformation
Serving
Storage
Analytics
ML/AI
Reverse ETL

Analytics (BI or Business Intelligence)

Dashboards

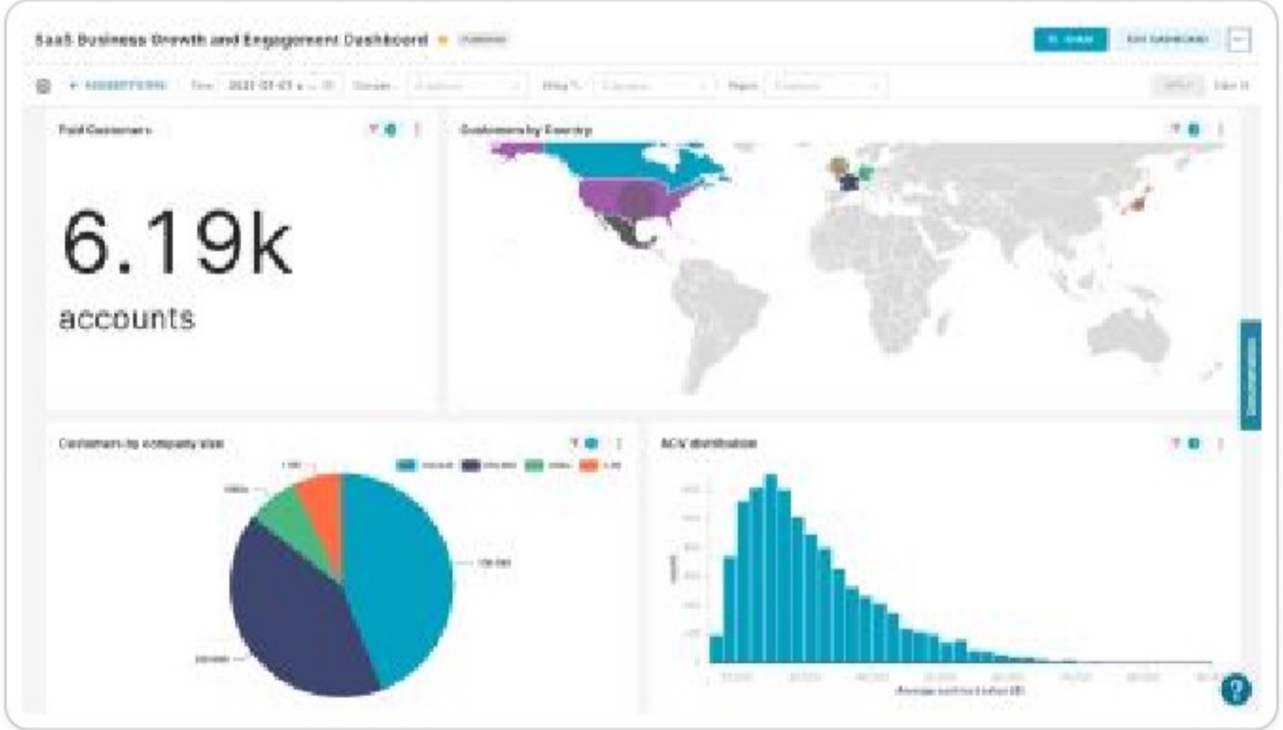
Reports

Ad hoc analysis

Looker Power BI

tableau

Superset



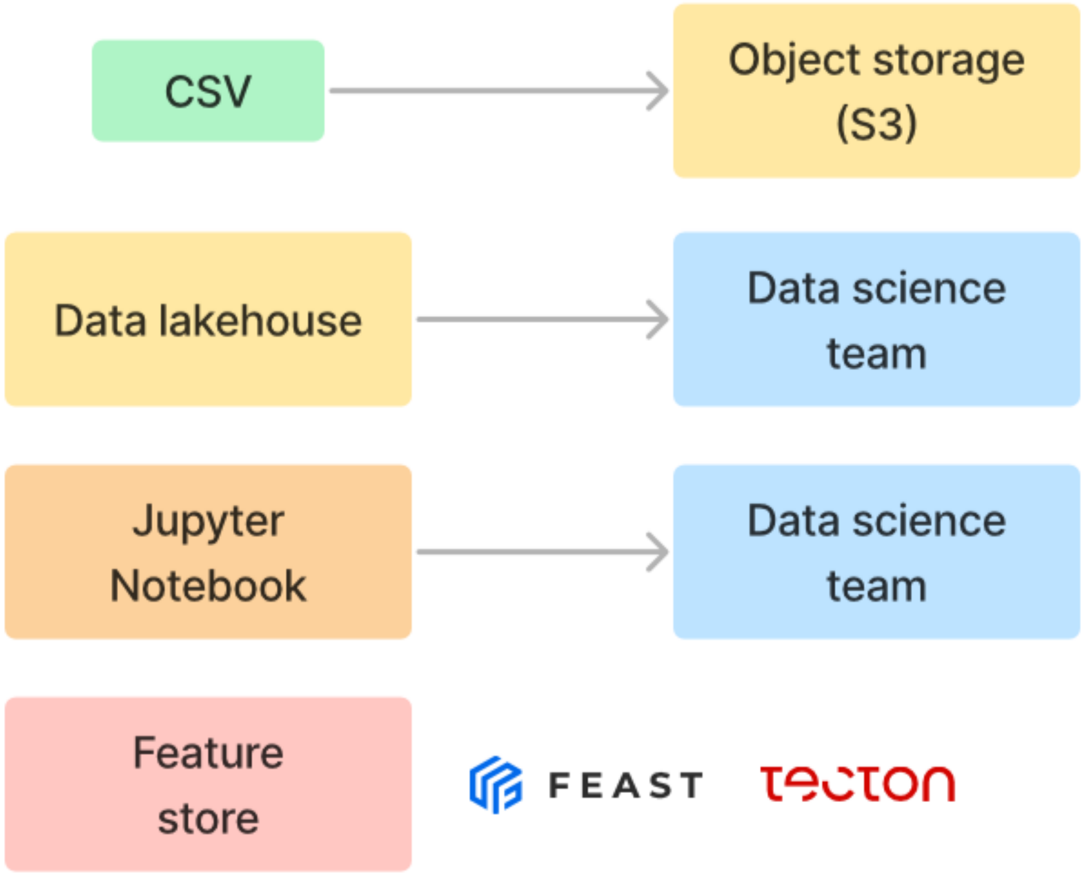
Think backward :

Why this metrics are useful?

Trust :

Trust is very hard to gain but easy to lose

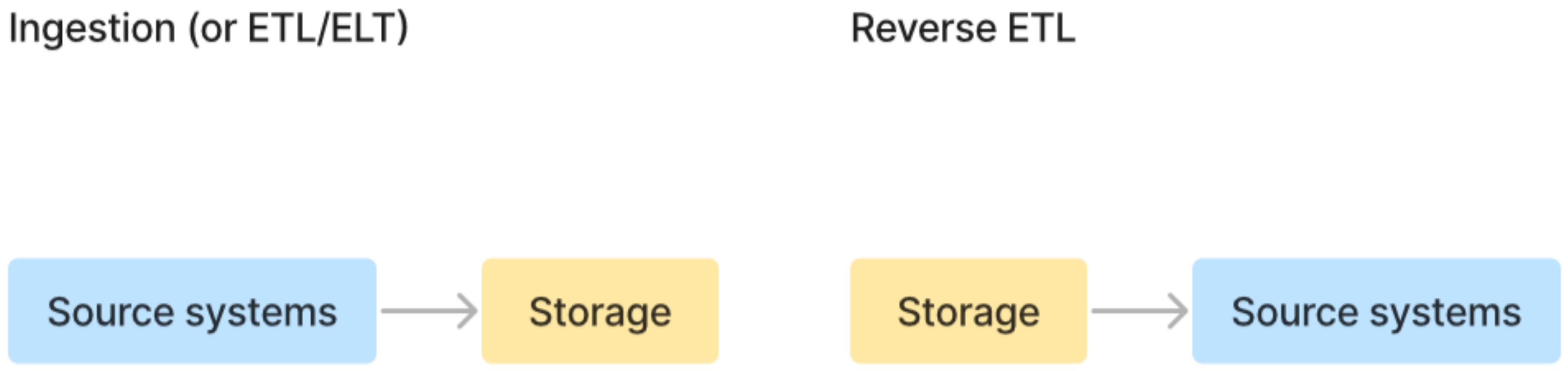
ML


CSV
Object storage (S3)
Data lakehouse
Data science team
Jupyter Notebook
Data science team
Feature store
FEAST
tecton

Deep Learning

ML	Deep Learning
Structured (tabular) data	Unstructured data - e.g. images
Limited # of features (100-1,000s)	Large # of features (millions)
Smaller datasets	Much larger datasets
Relatively small compute (CPUs)	Requires lots of compute (GPUs)

Reverse ETL


Ingestion (or ETL/ELT)
Source systems
Storage
Reverse ETL
Storage
Source systems

4 Undercurrents

DataOps

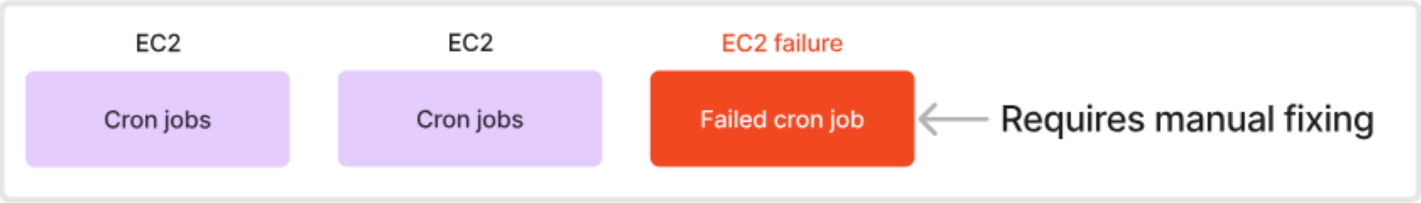
DevOps	DataOps (~DevOps for data)
Build/manage cloud infra	Build/manage cloud infra for data tools
Observability of cloud infra	Observability of data systems
Build automated CI/CD pipeline	Automation (including CI/CD)

Observability and monitoring

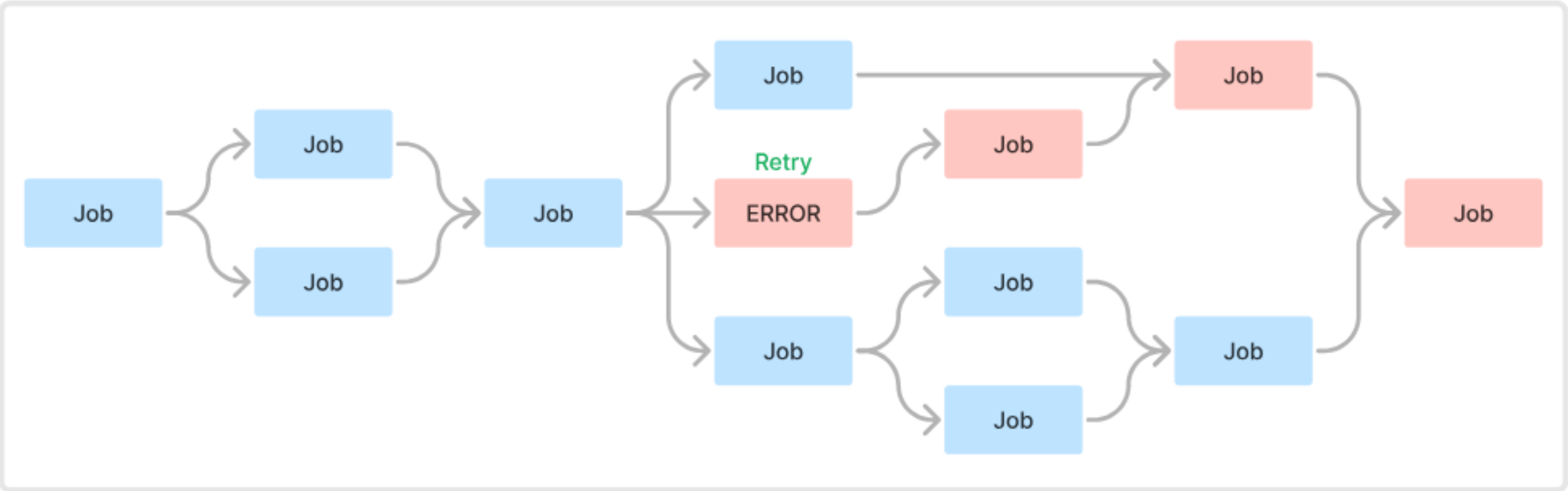
Data fails silently!

Automation

Cron jobs



Orchestration

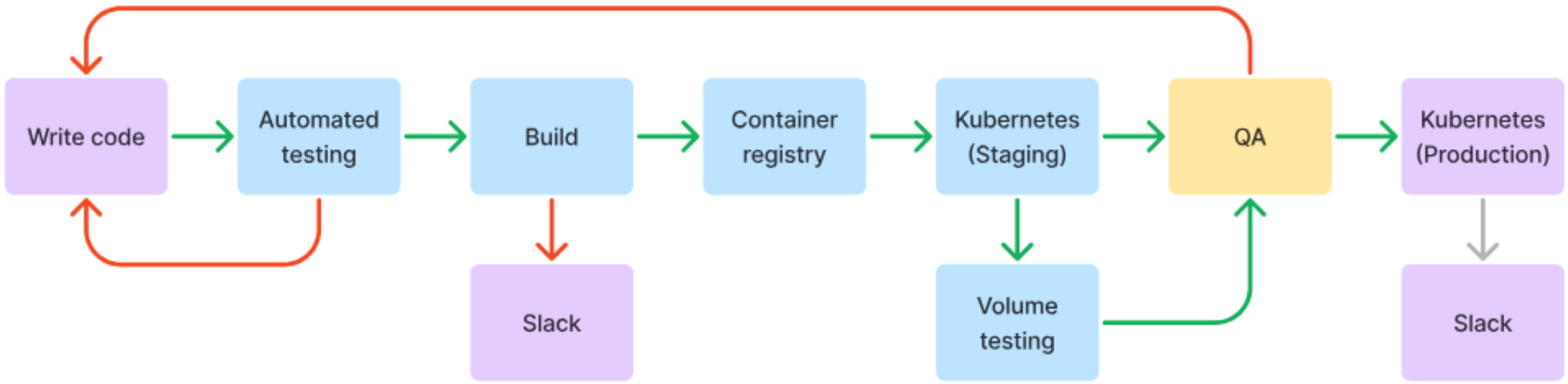


Automation - CI/CD pipeline

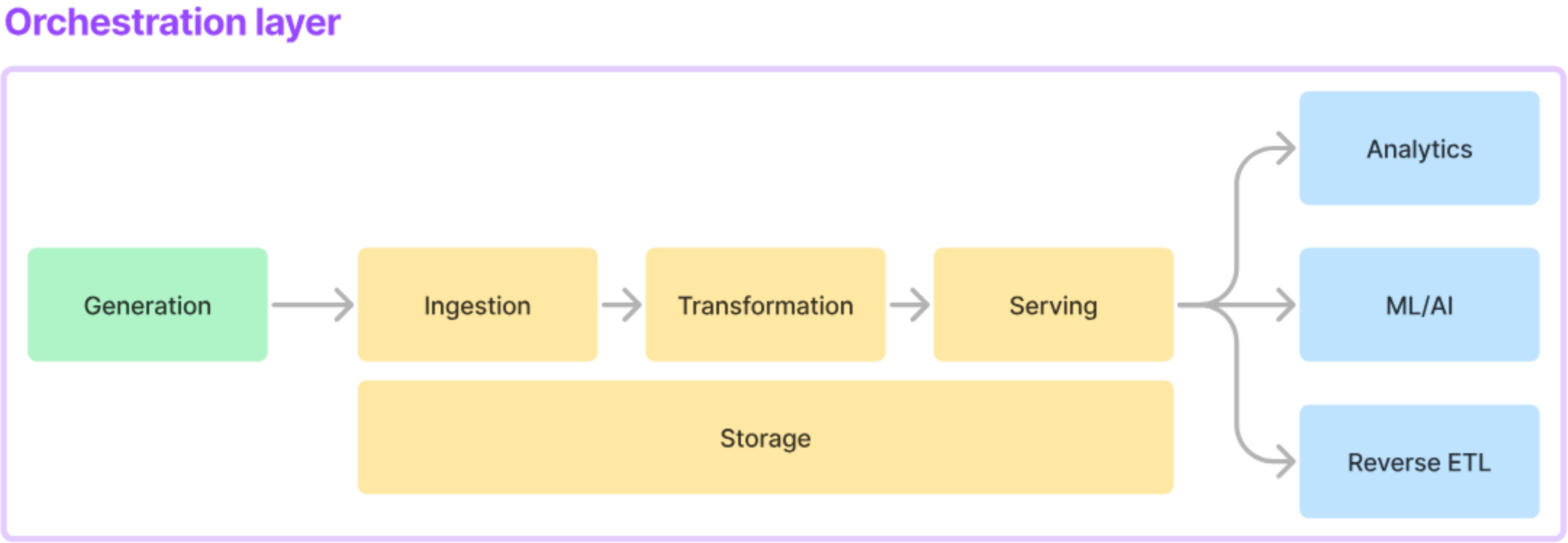
Most small teams overlook DataOps

Learn from software engineering

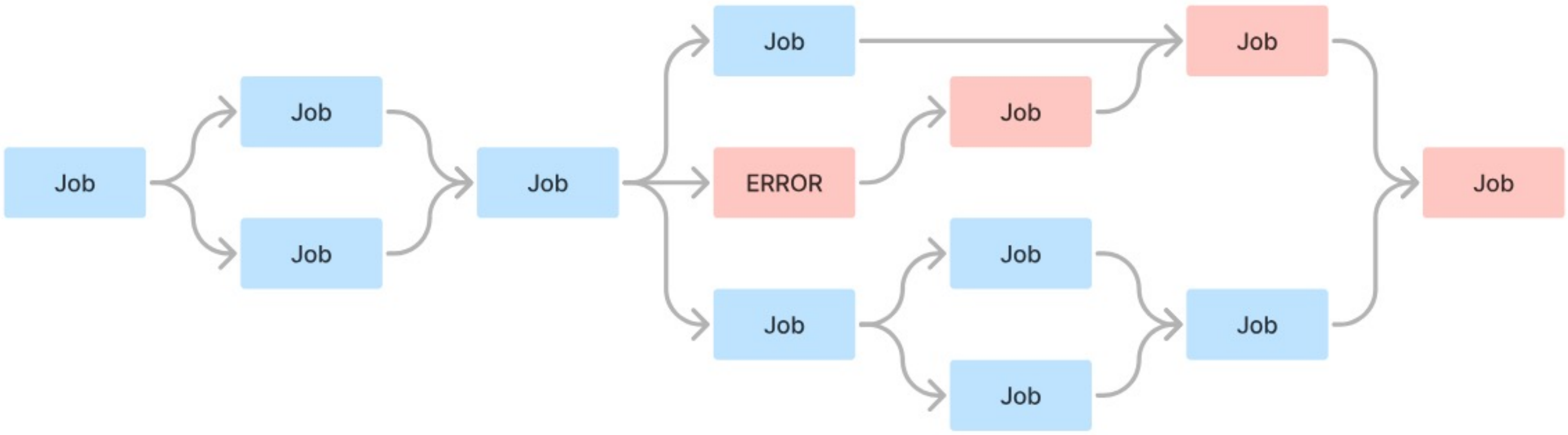
Example CI/CD pipeline



Orchestration



Orchestration - DAG



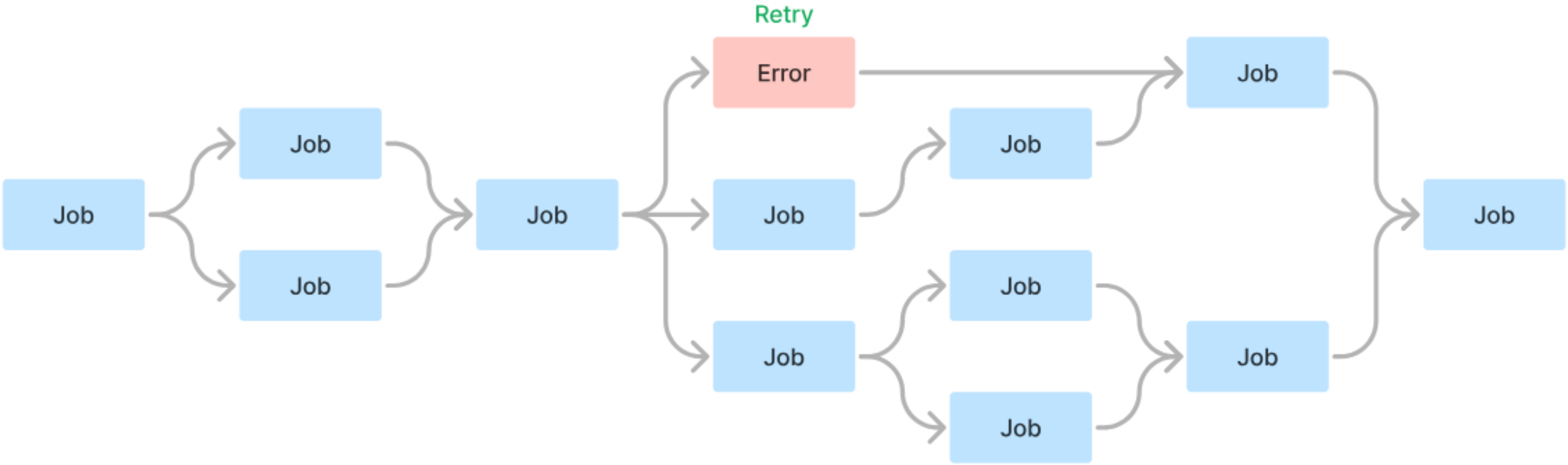
Why is orchestration so important?

Improve efficiency: Parallel executions

Complex dependency management

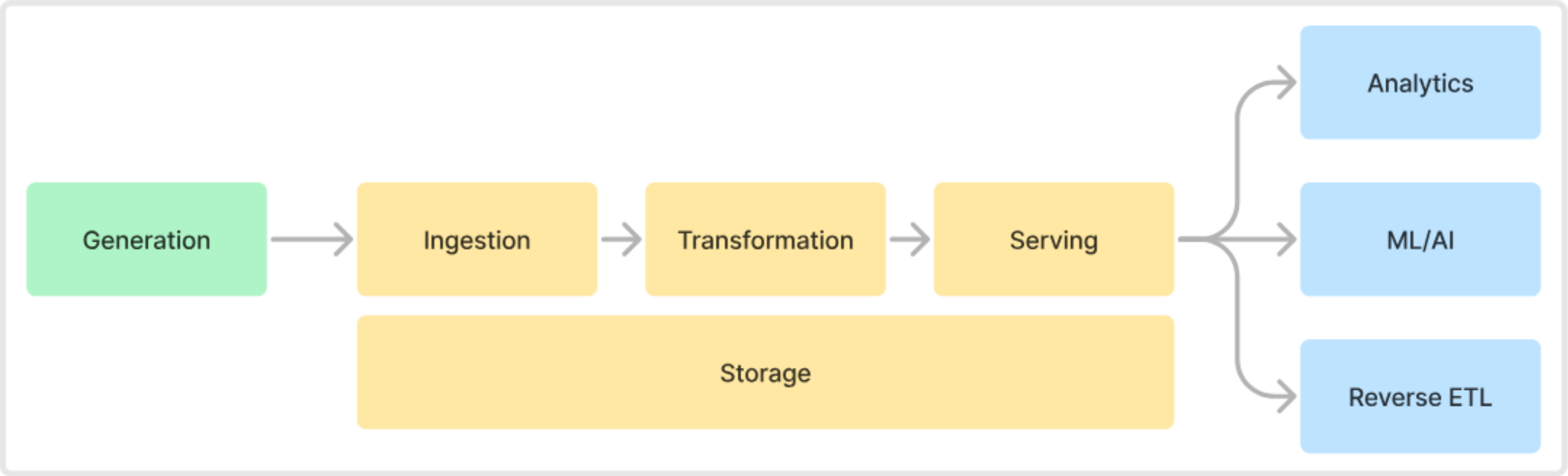
Improve reliability: Retries and timeouts

Automation



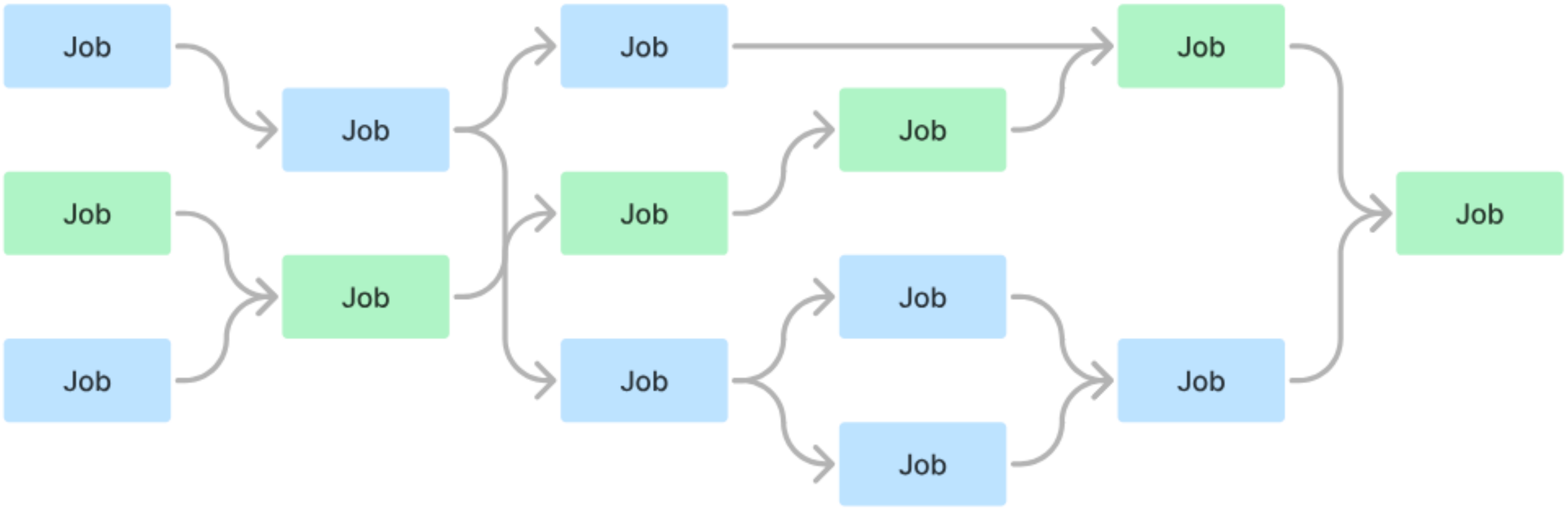
Why is orchestration so important?

End-to-end visibility



Why is orchestration so important?

Data lineage



Orchestration tools

Apache Airflow PREFECT dagster Flyte

Other undercurrents

Security

- Authentication
- Authorization or “Access Control”
- Principle of least privilege

Privacy

- Coming into focus
- High profile breaches
- Personally identifiable information (PII)

Data quality

More nuanced
than
it seems at first

It's all about
TRUST!

Data fails
silently

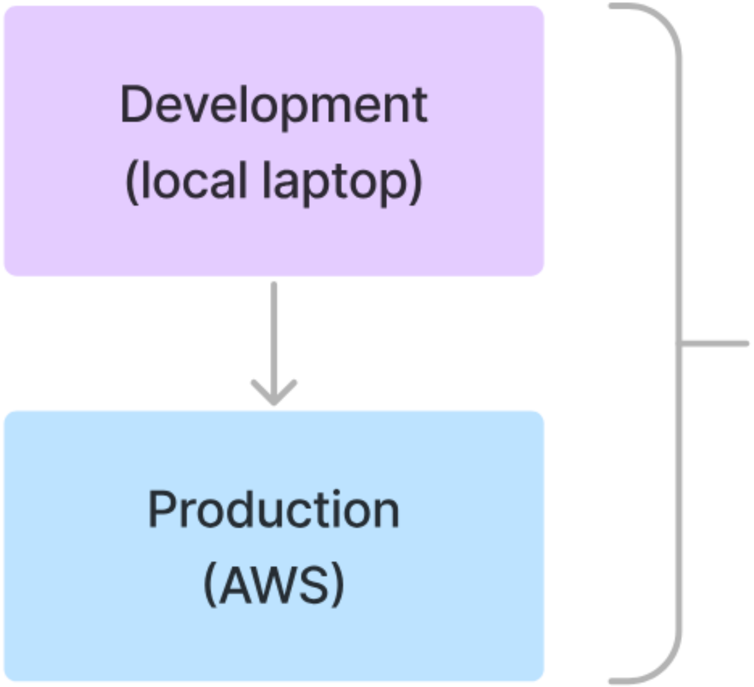
Validation
Testing
Observability

Data quality

Software problems only happen when you deploy software, but data problems can happen independently from code

Data testing is different from software testing - it's trickier

Development workflow



Benefits

- Fast iteration
- Isolation (safe to tinker without affecting prod env)
- Cost reduction (use local laptop for dev)

Challenges

- Sampling a large dataset is not always easy
- Dev/prod environment parity not always possible
- Cloud tools not always available locally

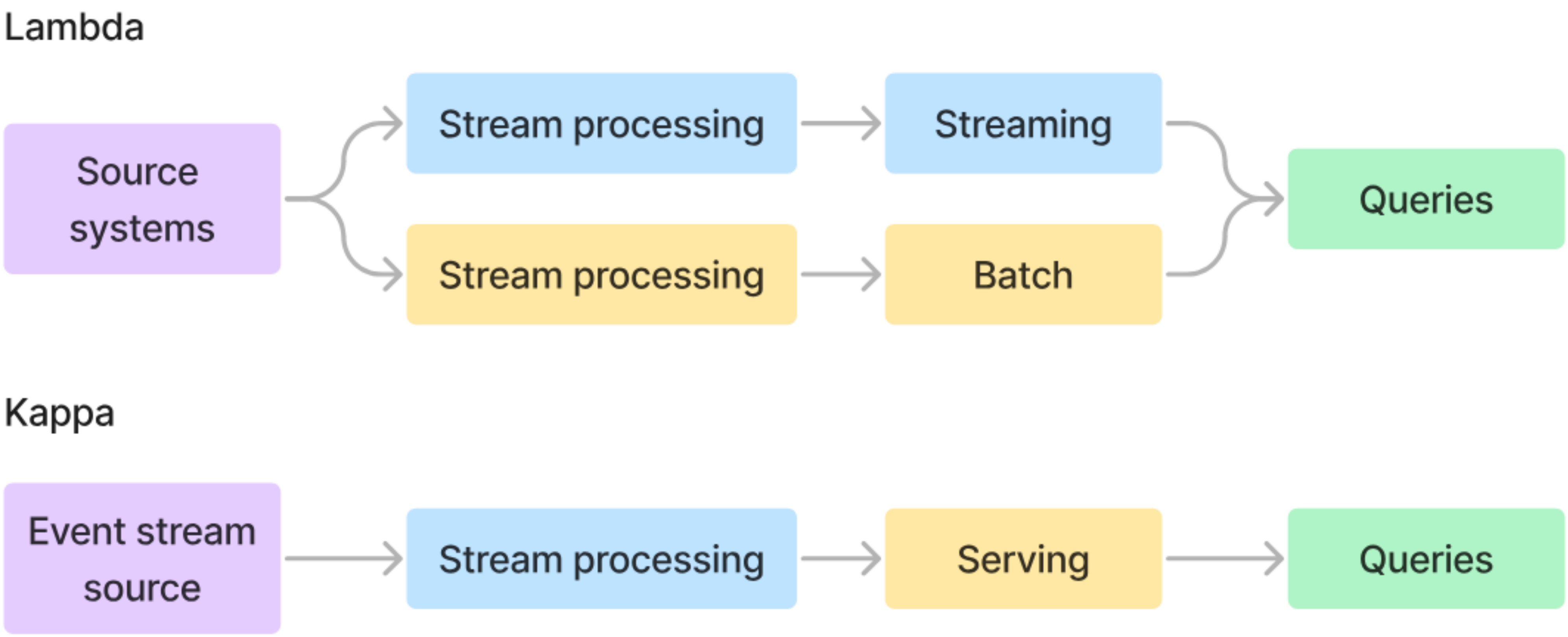
5

Data Architecture

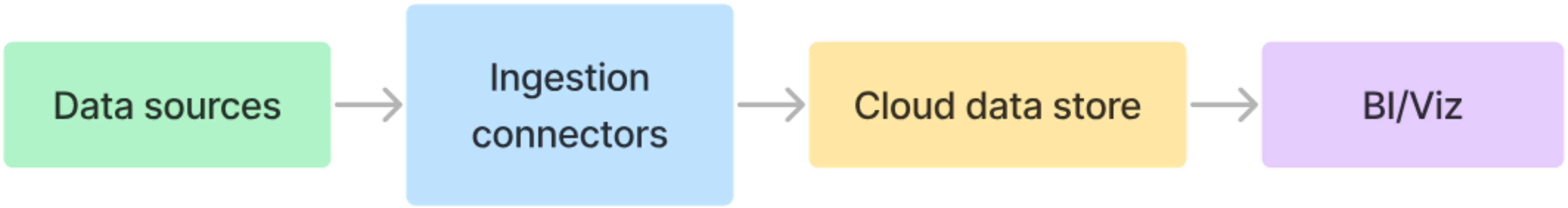
What is a good data architecture?

Performance	Scalability	Reliability	Expect failure
Security	Modularity	Cost-effectiveness	Flexibility

Lambda & Kappa architectures



Modern data stack



What factors should you consider when designing your data architecture?

Speed

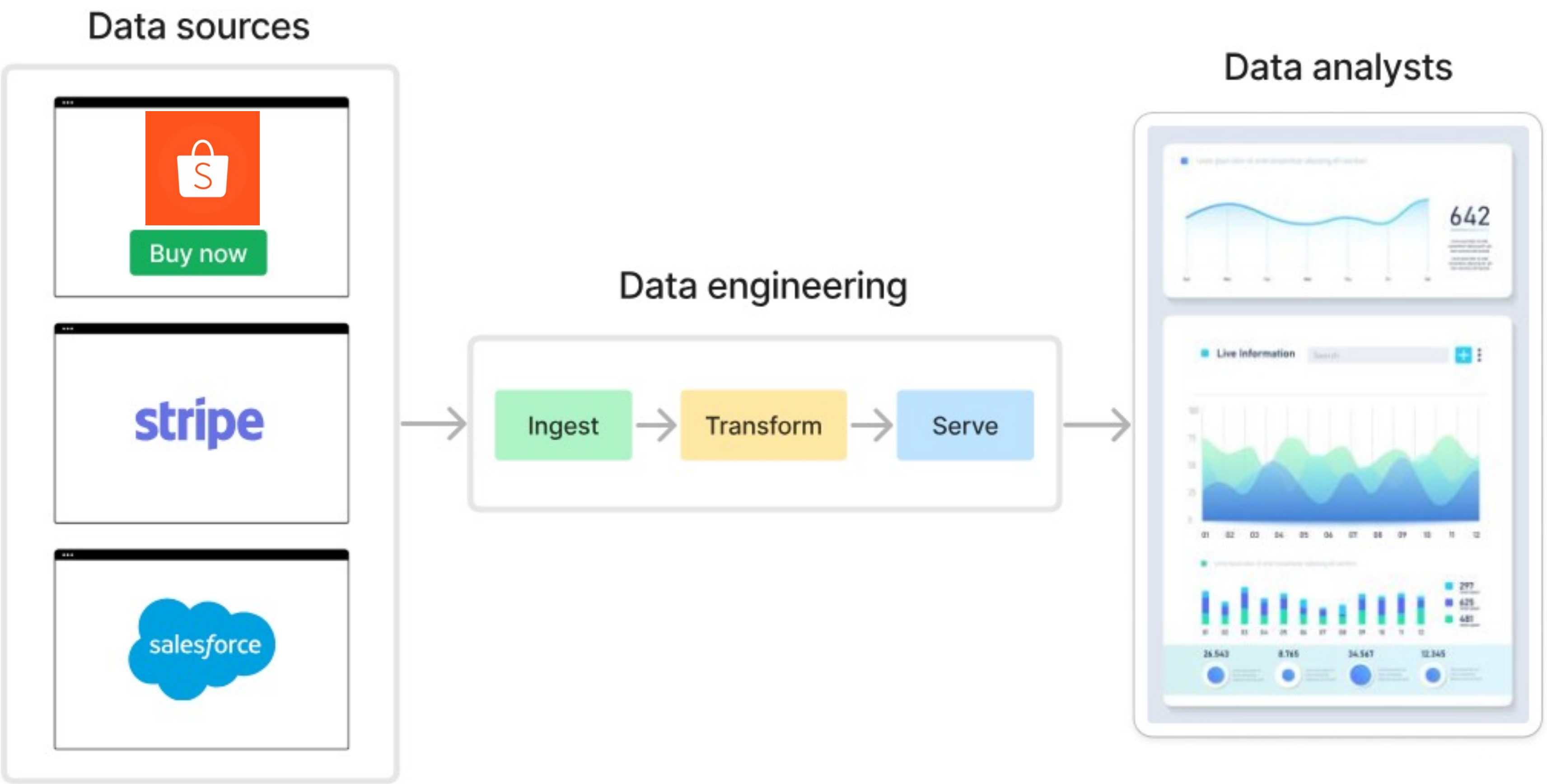
Size of the team

Integration

Start with tools central to the pipeline and that are hard to change

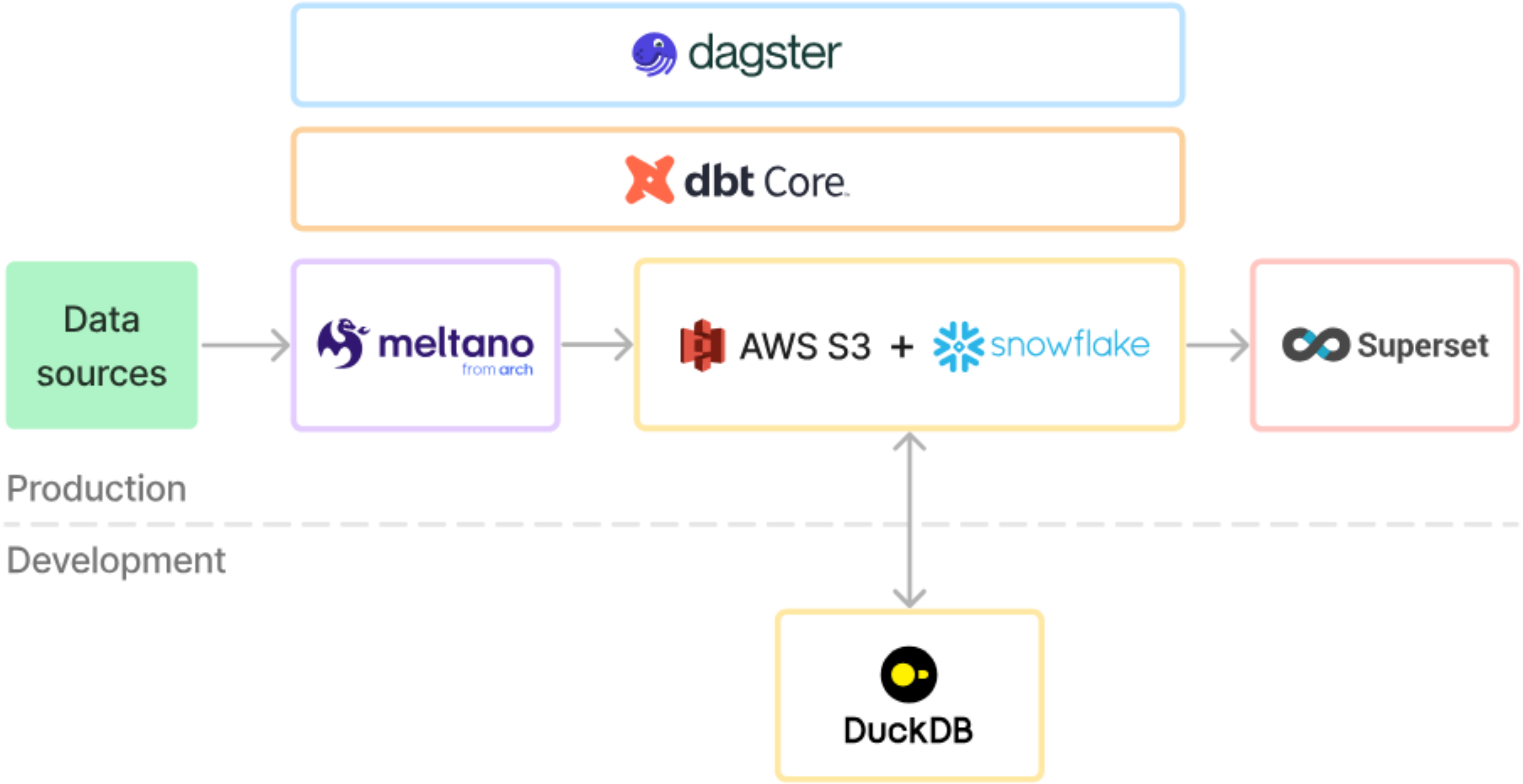
One caveat:
Be skeptical of benchmarks

BI Stack

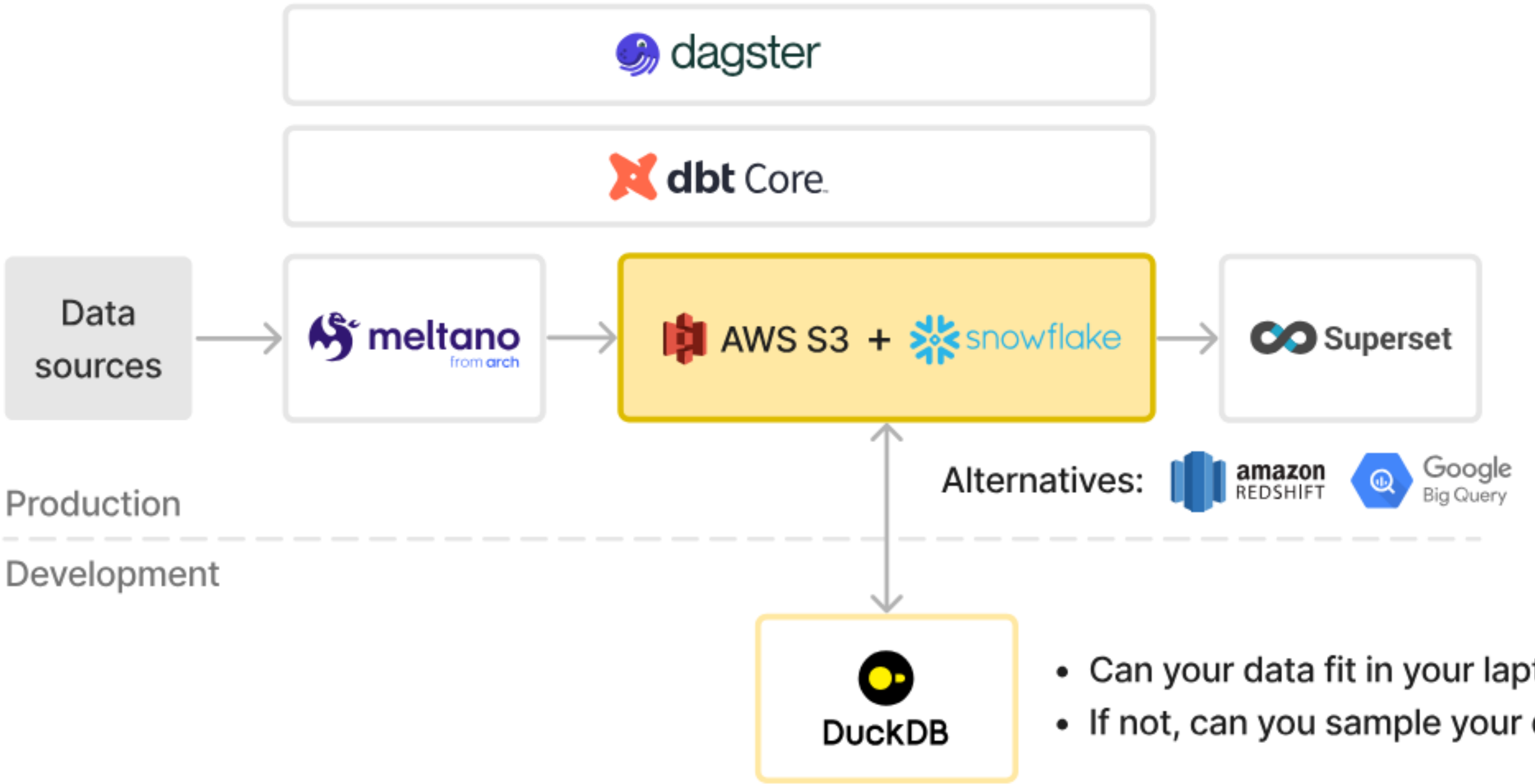


Data sources
Buy now
stripe
salesforce
Data engineering
Ingest
Transform
Serve
Data analysts
642
Live Information

BI stack v1 - Data warehouse

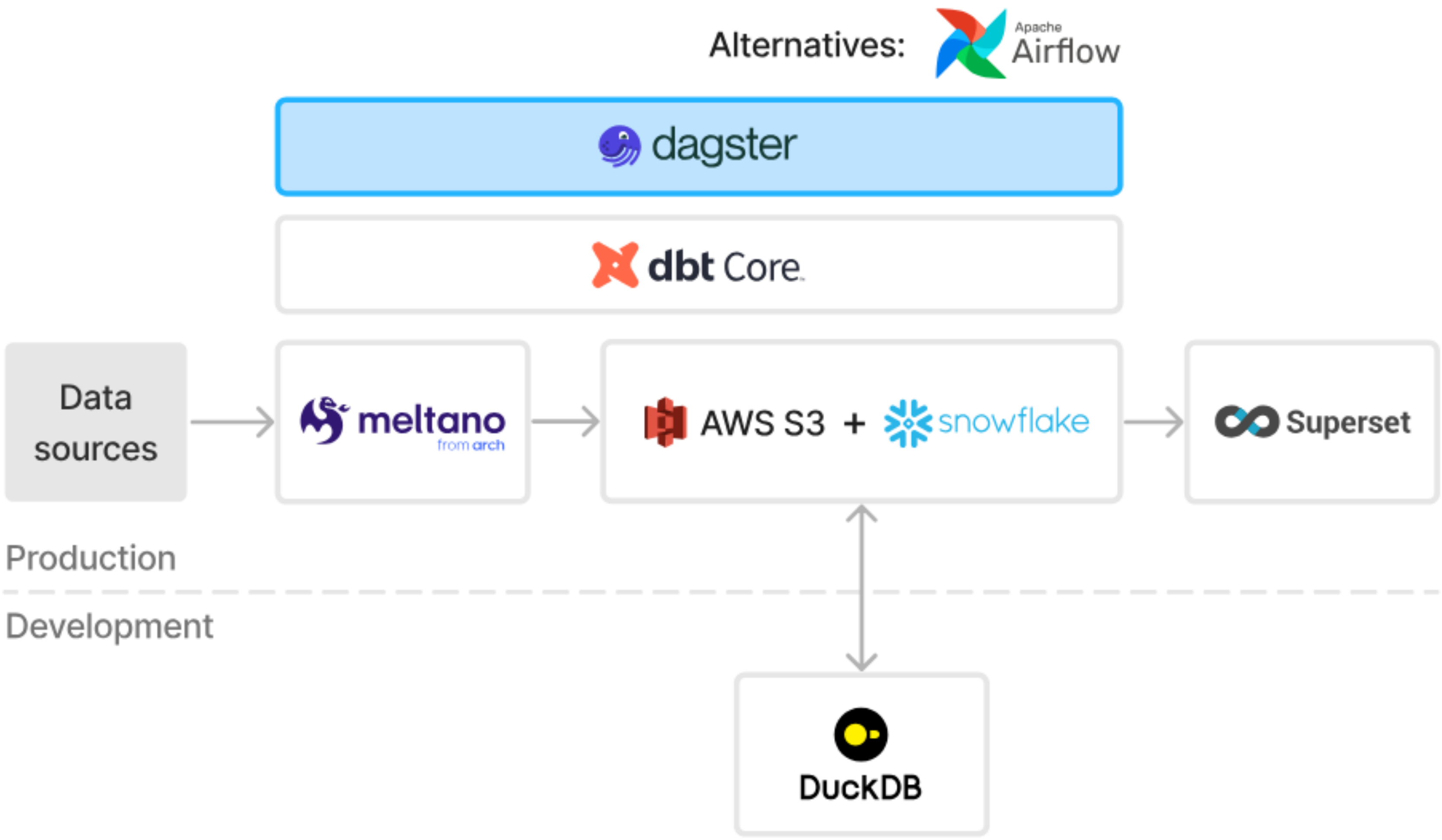


Central storage



- Can your data fit in your laptop?
- If not, can you sample your data?

Orchestrator



Orchestrator - Airflow vs Dagster

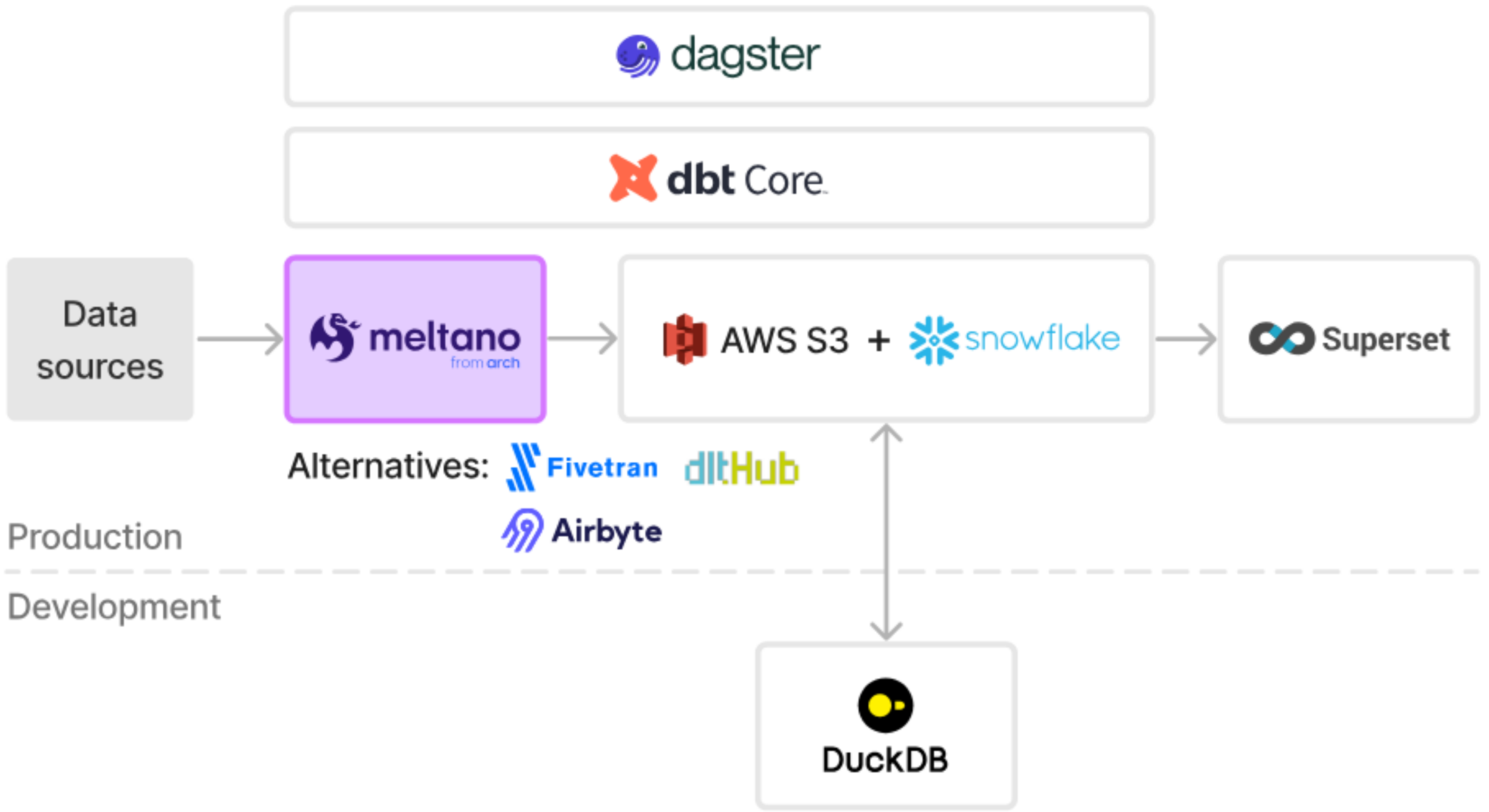
Apache Airflow	dagster
No local env setup	Local dev env
Dependency management issues	Cloud-native, easy deployment, CI/CD
Complex to deploy and manage	Declarative (rather than imperative)

Imperative vs declarative

Imperative	Declarative
The “how” Spell out the exact steps	The “what” Specify the desired outcome

ReactJS kubernetes

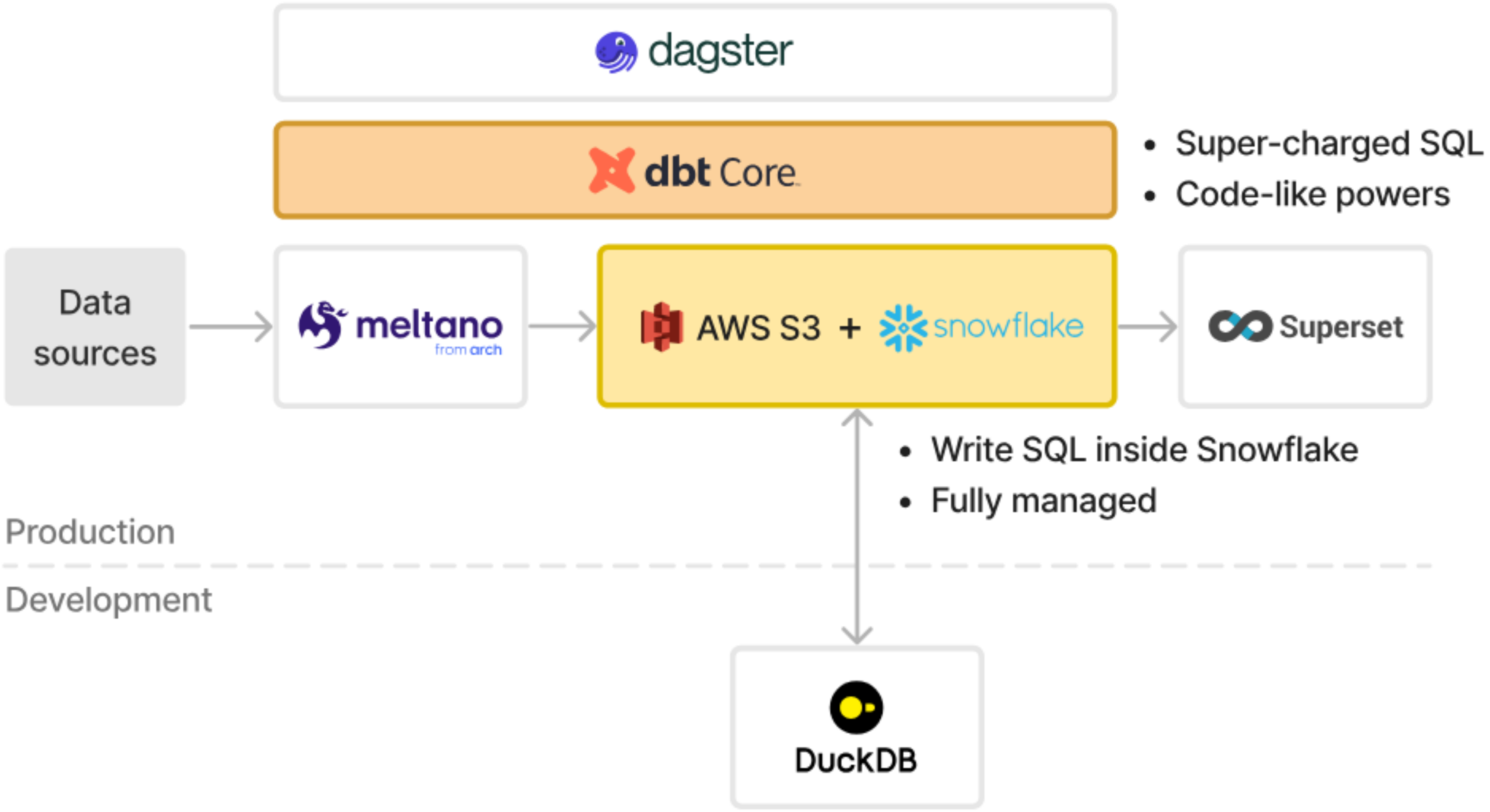
Ingestion


dagster
dbt Core
Data sources
meltano
from arch
AWS S3 + snowflake
Superset
Alternatives: Fivetran dltHub
Airbyte
Production
Development
DuckDB

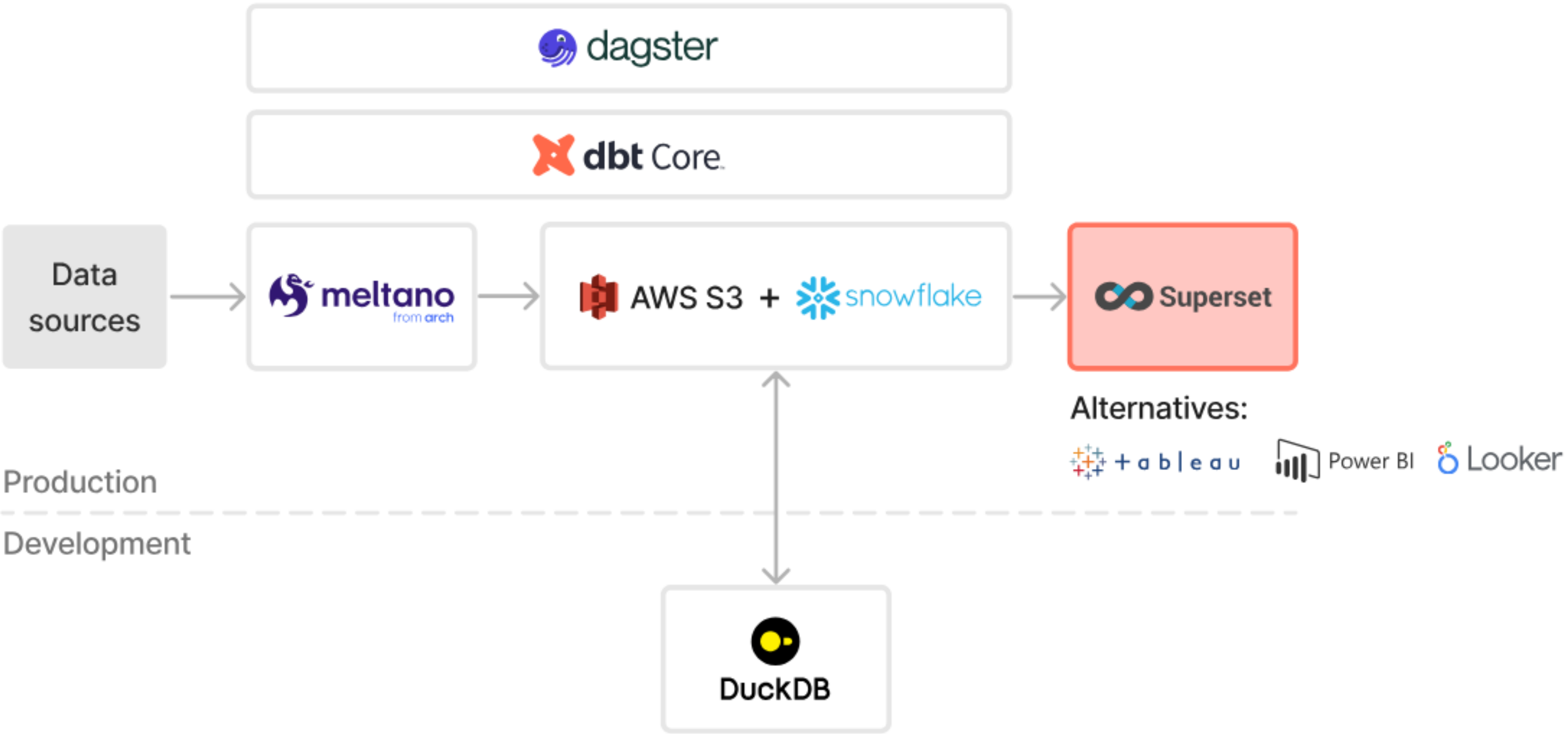
Ingestion tool comparison

	Fivetran	Airbyte	meltano from arch	dltHub
Open-source?	❌	✓	✓	✓
UI or code-first	UI	UI	Code	Code
# of connectors	100s	100s	100s	<20
Embedded?	❌	❌	❌	✓

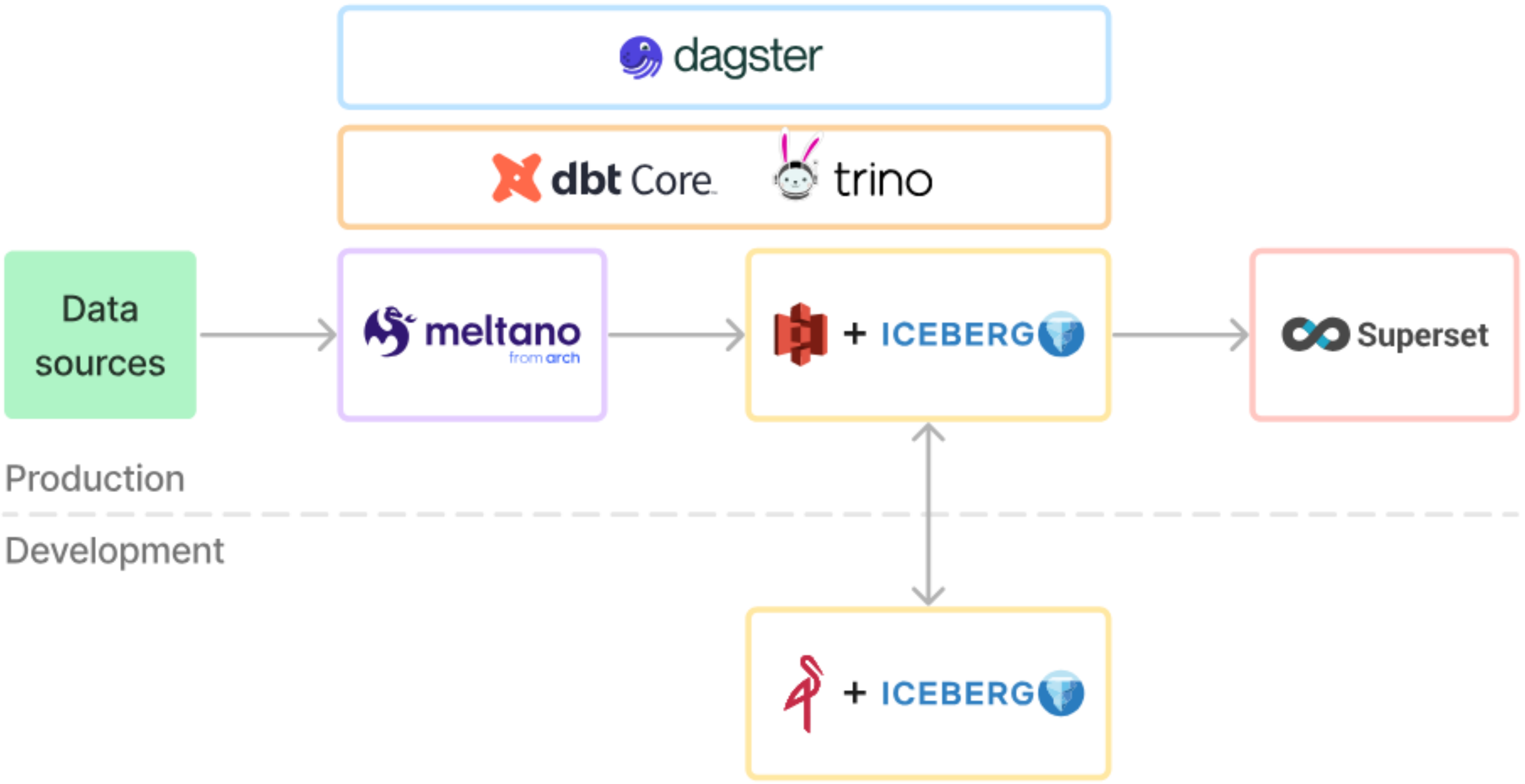
Transform


dagster
dbt Core
Super-charged SQL
Code-like powers
Data sources
meltano
from arch
AWS S3 + snowflake
Superset
Write SQL inside Snowflake
Fully managed
Production
Development
DuckDB

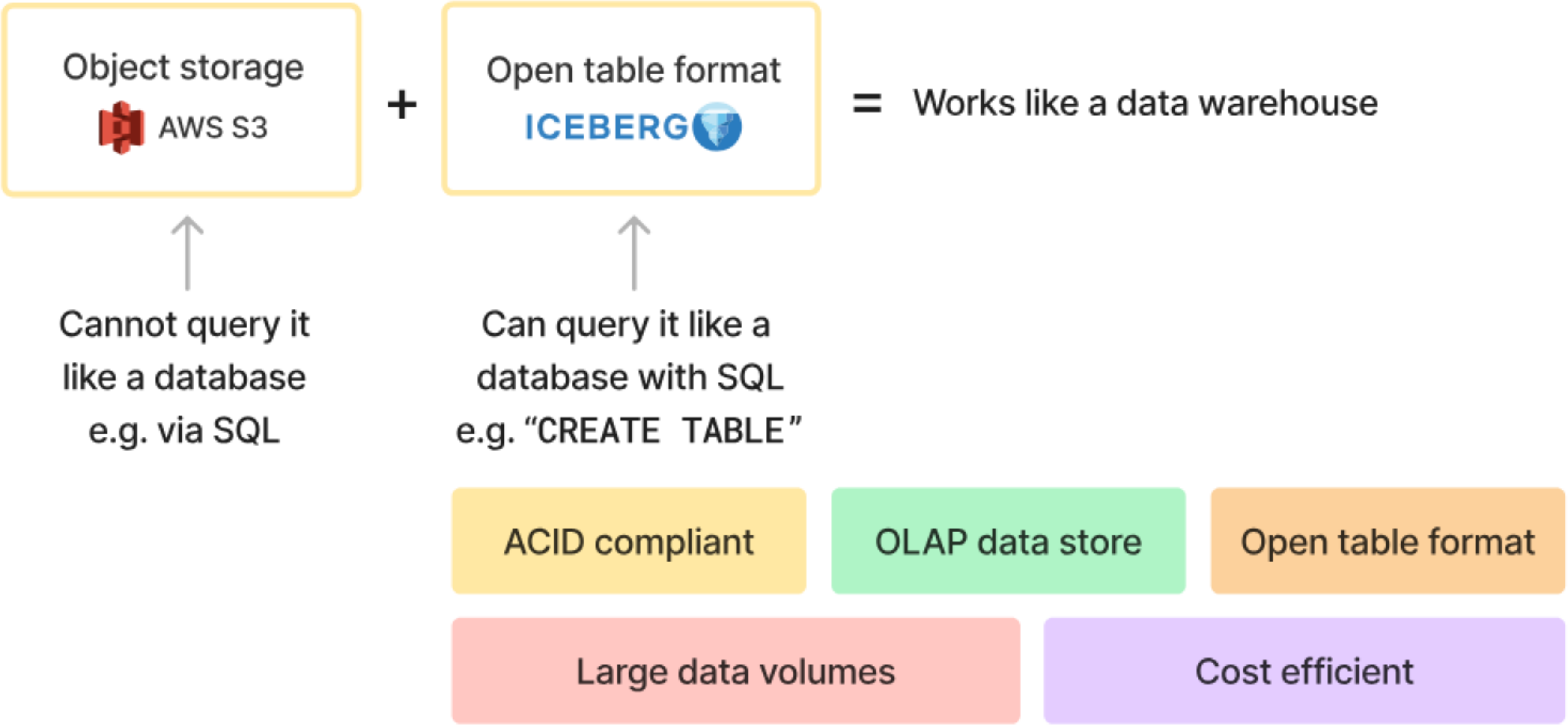
Visualization



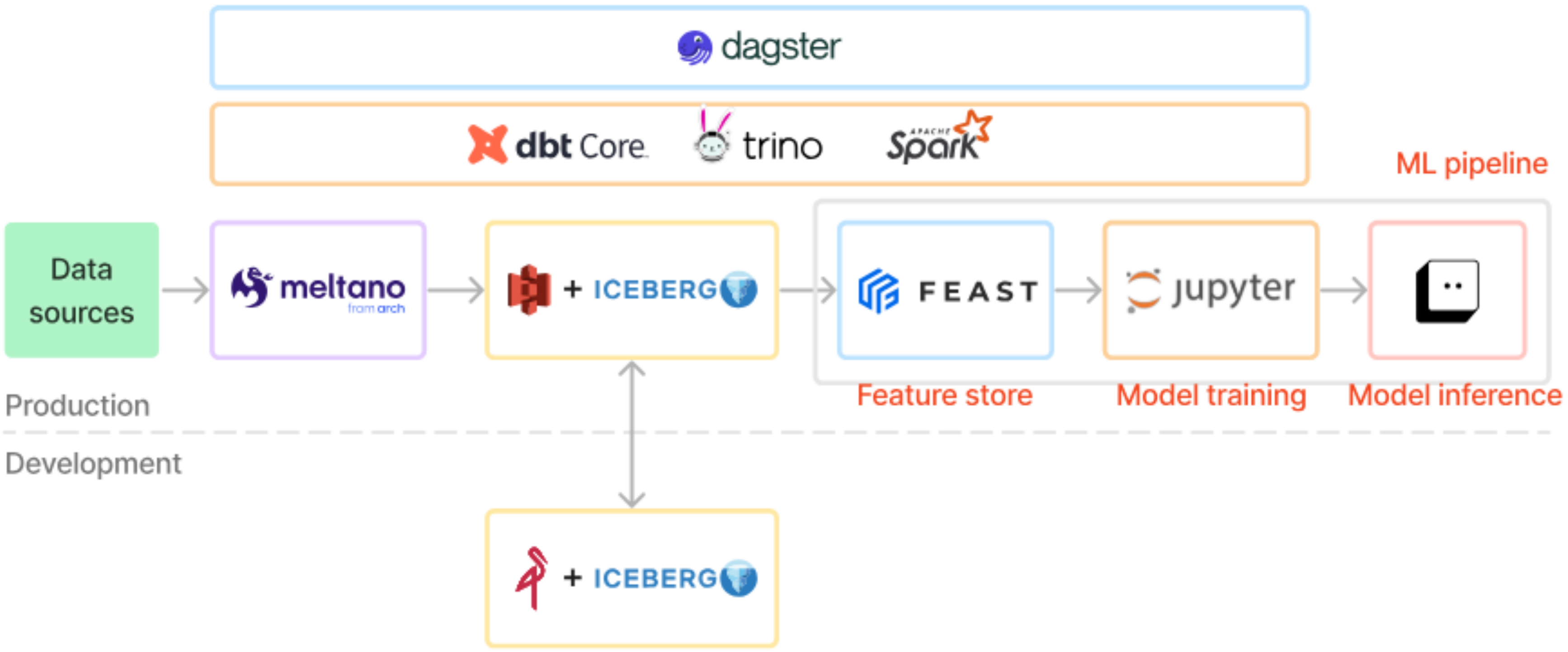
BI stack v2 - Data lakehouse



Apache Iceberg



ML stack


dagster
dbt Core
trino
Apache Spark
ML pipeline
Data sources
meltano
from arch
+ ICEBERG
FEAST
jupyter
Feature store
Model training
Model inference
Production
Development
+ ICEBERG

Model training

Model training

Experiment tracking

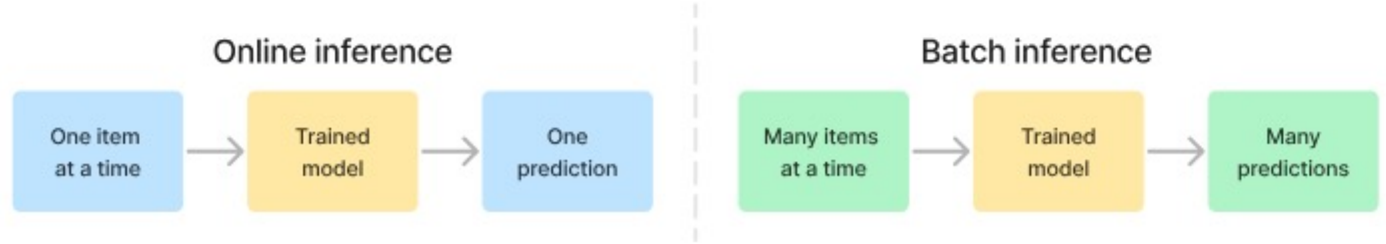
Data lineage / Data versioning

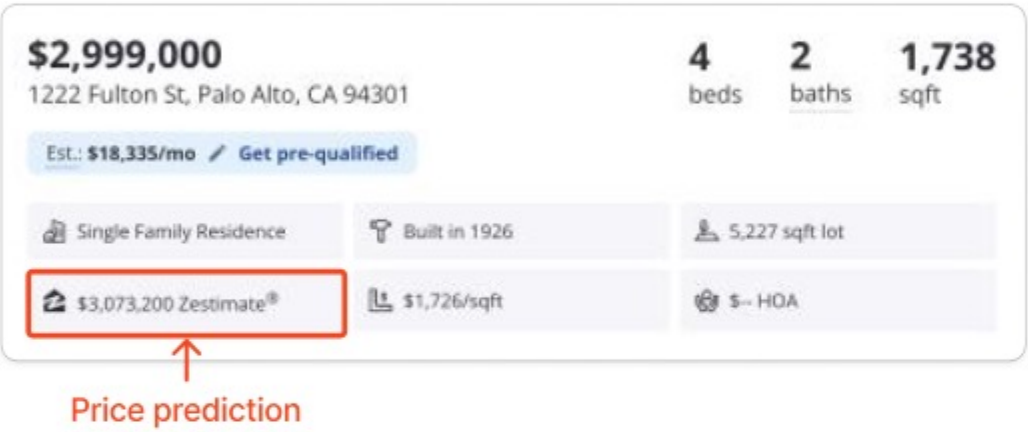
jupyter RAY

mlflow Weights & Biases

dagster ICEBERG

Model serving



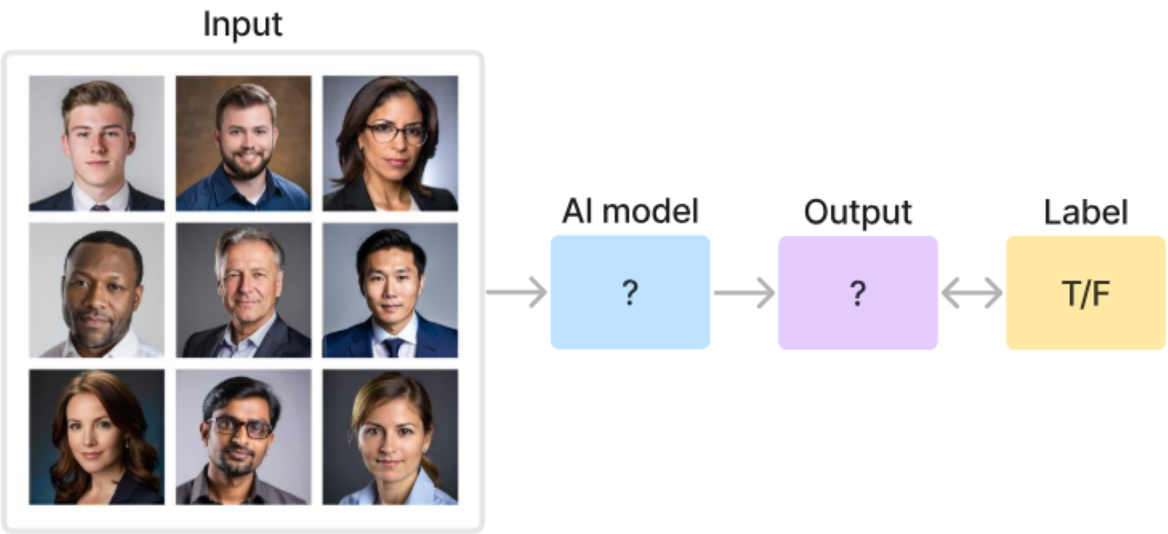


Tools:

BentoML

Labeling

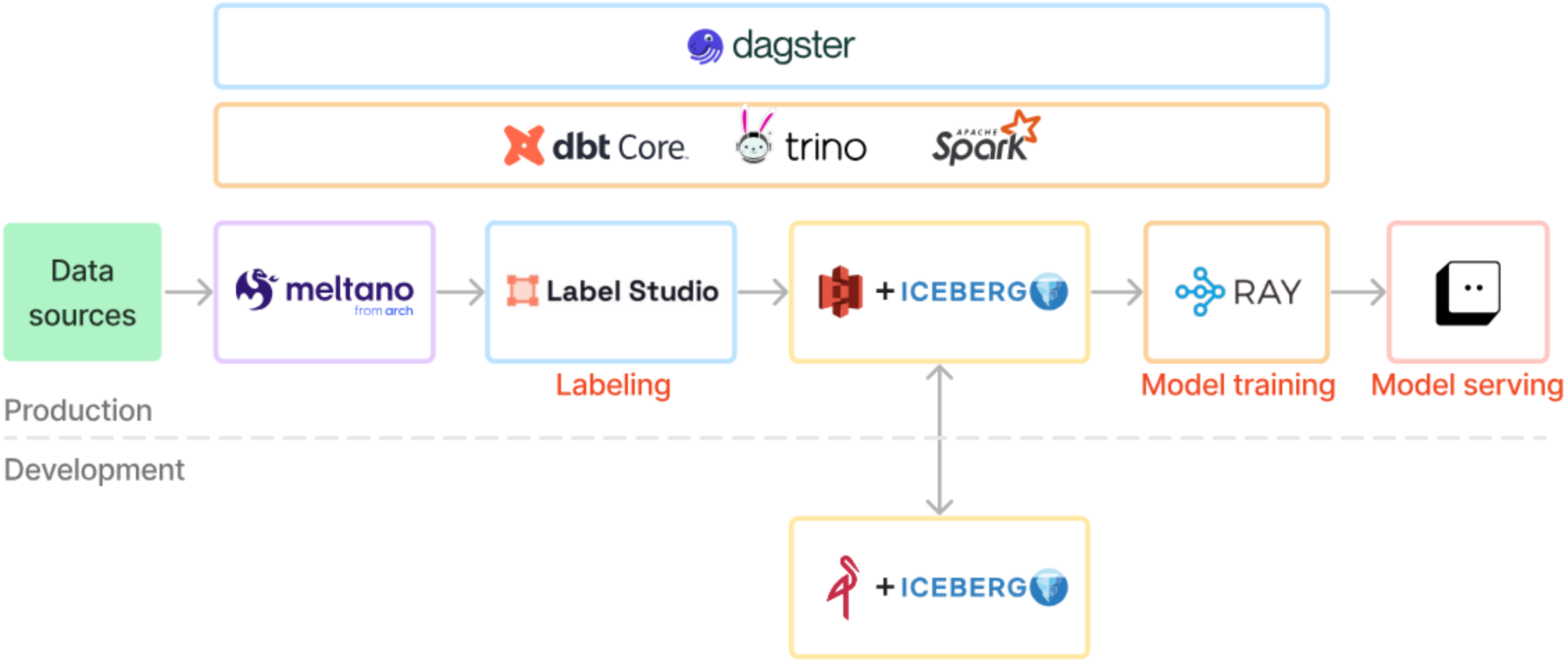
For training the AI model



Dataset (human labeled)

id	img_url	label
1	s3://john.jpg	True
2	s3://lisa.jpg	False
...	...	...
10000	s3://blah.jpg	True

Deep Learning stack



LLMs?

OpenAI

Retrieval Augmented Generation

Vector databases

Other considerations

Data validations

- Type validation
- Constraint validation
- Code validation

dagster, dbt Core, great expectations

Testing

- Unit testing
- Statistical testing

dagster

Observability

- Monitoring, alert, notifications
- Upstream schema changes
- Detailed logs and stack traces
- ALWAYS expect failures
- Observe infra, not just data pipeline

CI/CD

- Automate CI/CD process
- Automation = more frequent releases

Terraform, Pulumi, Github Actions

Great ! We’ve just finished the overview of

Data Engineering