[639] Data Ingestion

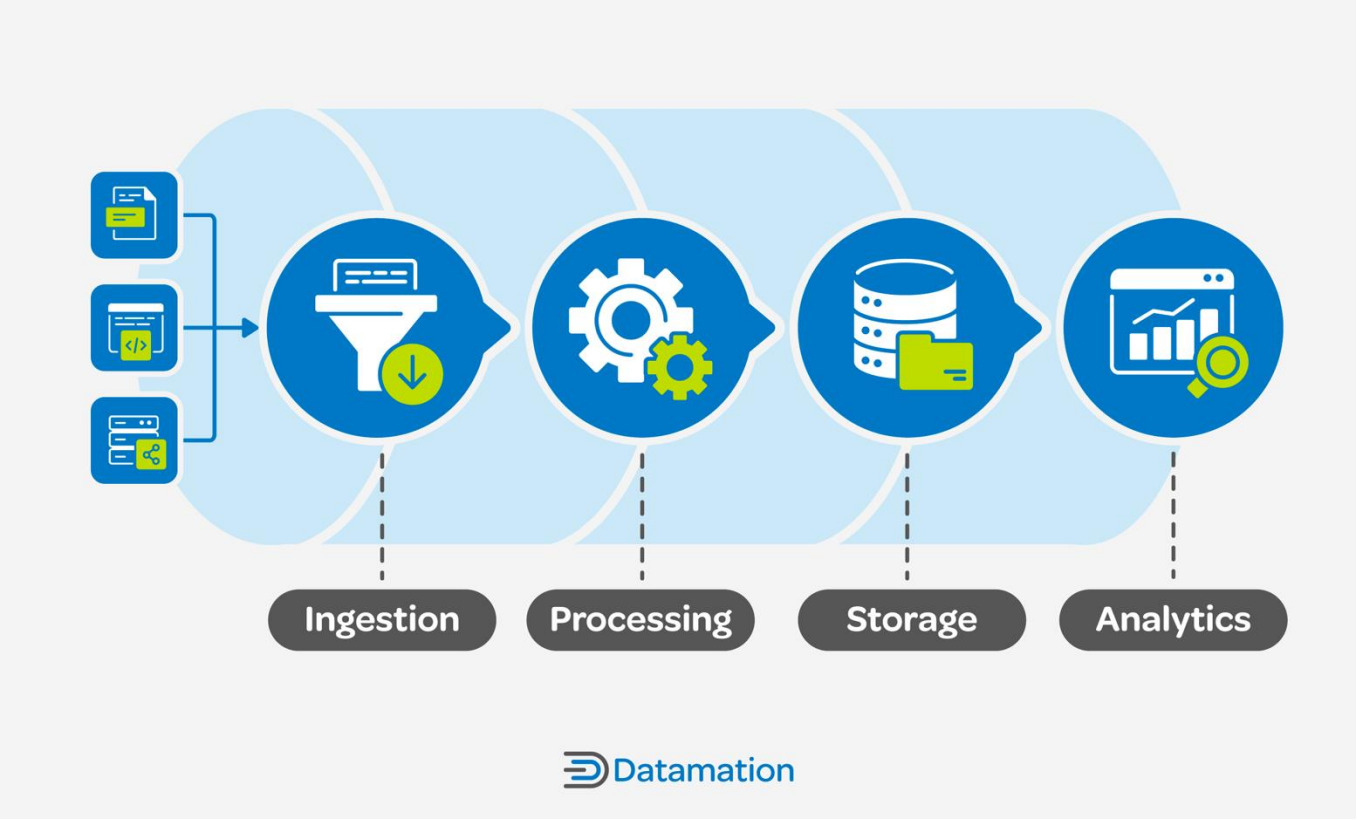
Meenakshi Syamkumar

Learning Objectives

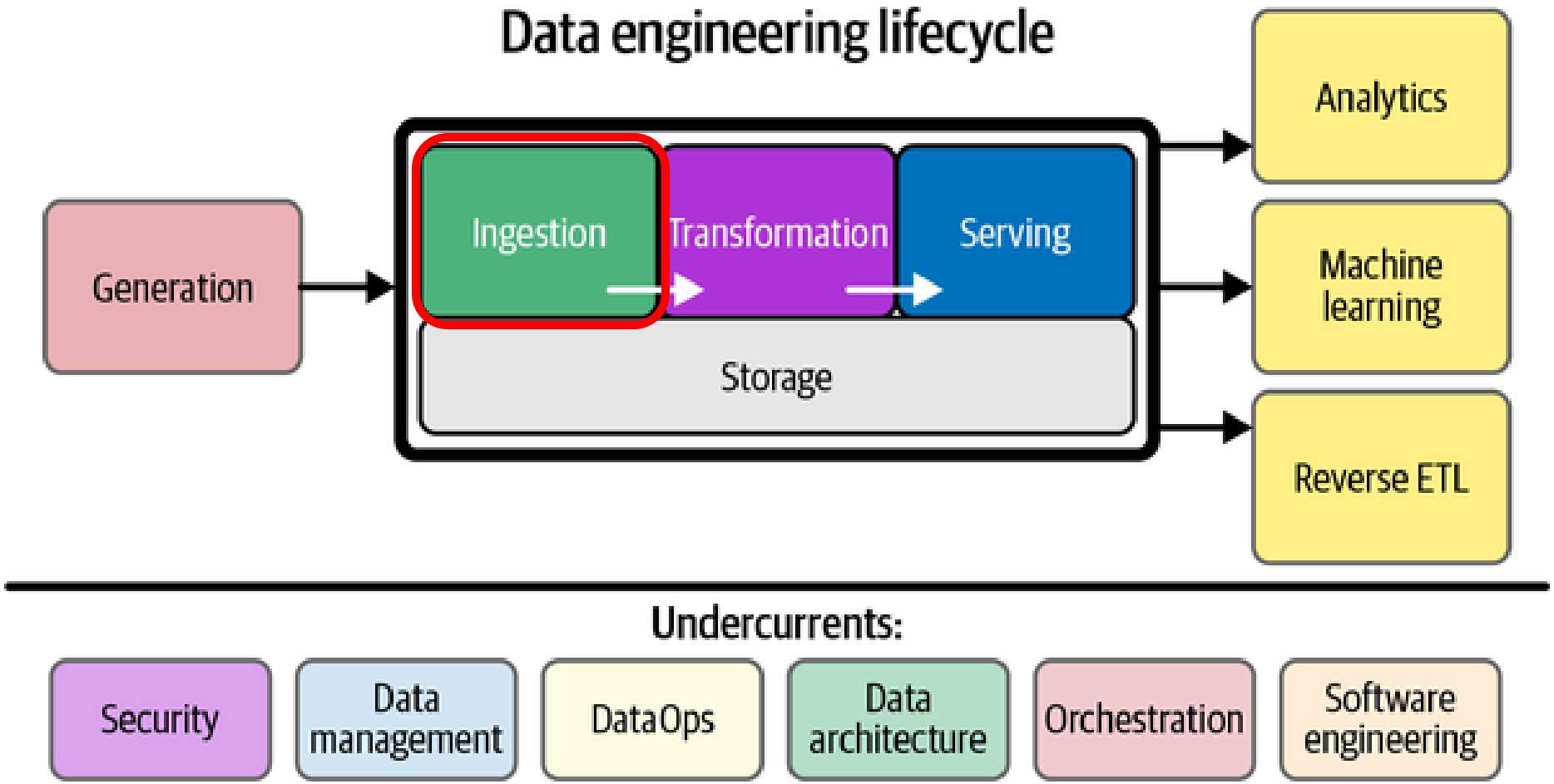
Define	data pipelines and data ingestion
Understand	the key considerations that govern data ingestion design

What are data pipelines?

- Combination of architecture, systems, and processes that move data through the stages of data engineering lifecycle
- Key components of data pipelines:
  - Data ingestion (from source systems)
  - Data processing (aka integration)
  - Data storage
  - Data analysis and visualization



Data Engineering Lifecycle



What is data ingestion?

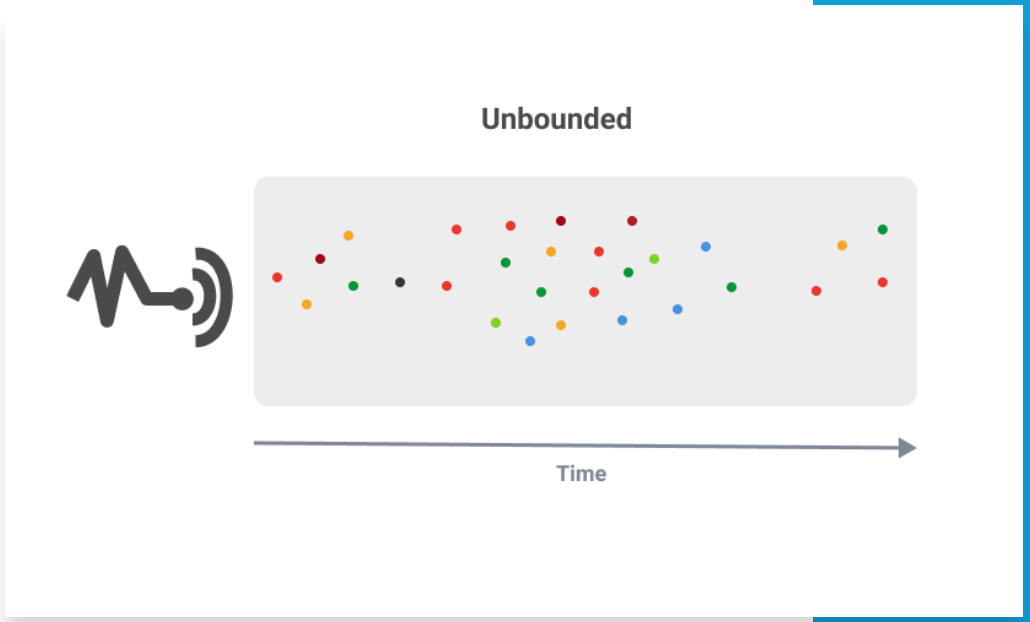
- Process of moving data from one place to another
- Data movement from source systems into storage

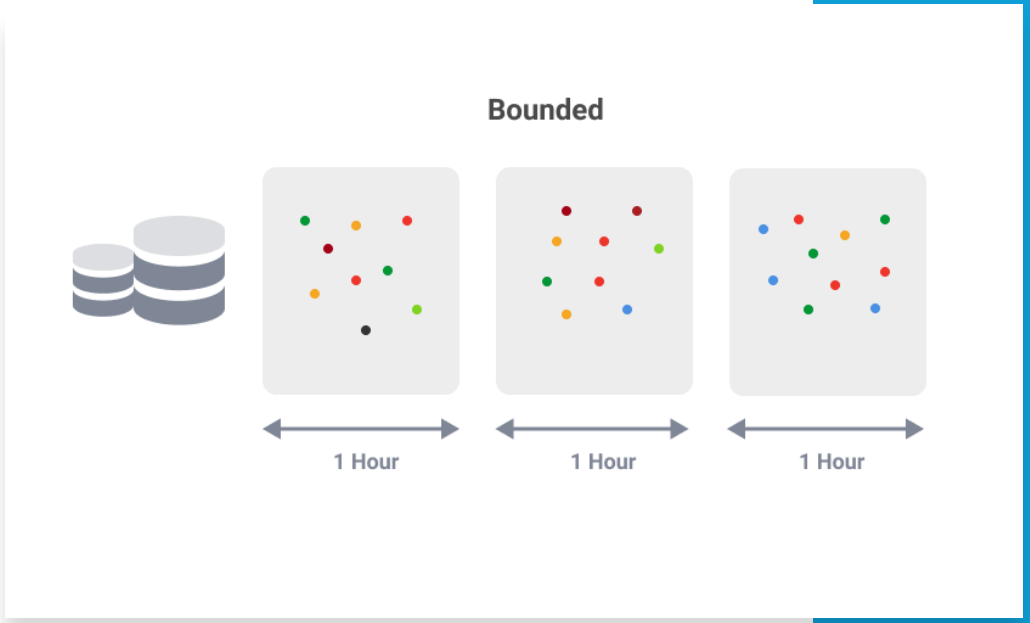
Key considerations for Data Ingestion

- What is the use case for the data being ingested?
- Can the data be reused instead of ingested multiple versions?
- Where is the data's destination?
- How often should data be updated from the source?
- What is the expected data volume?
- What format is the data? Can the downstream storage and transformation accept this format?
- Is the data of good quality for immediate downstream use?
- Does the streaming data require in-flight processing for downstream ingestion?

Bounded vs Unbounded data

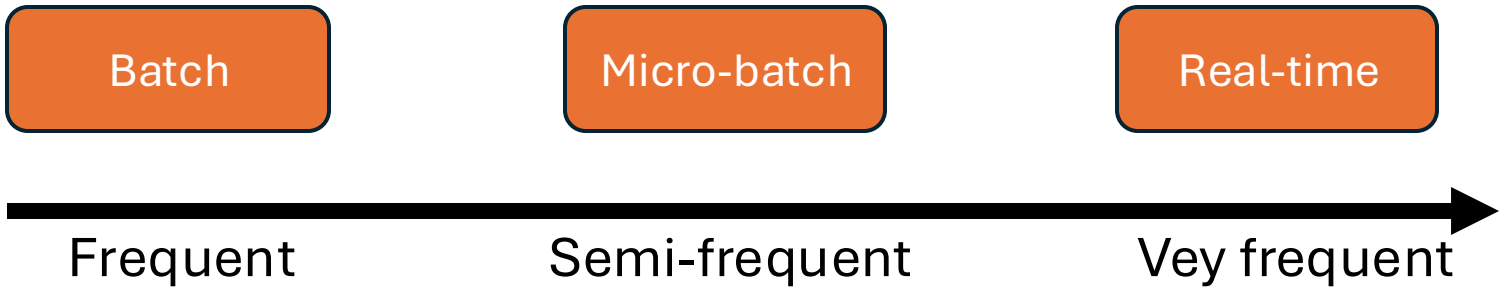
- Unbounded data: data exists as events happen, either sporadically or continuously, ongoing, and flowing
- Bounded data: convenient way of bucketizing data across some sort of boundary such as time





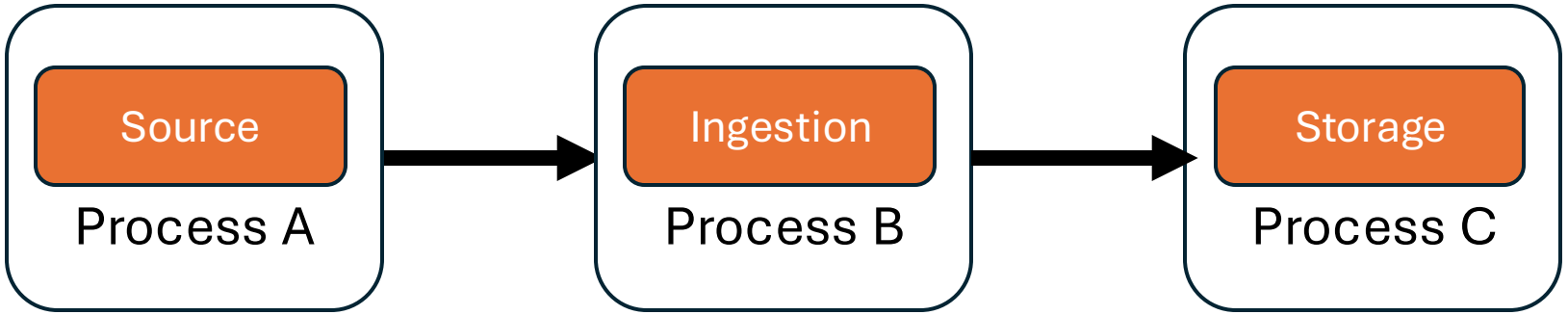
Frequency

- “Real-time” ingestion aka streaming patterns are becoming increasingly common
  - Streaming systems are a good fit for many data sources
  - Ex: IoT systems sensor’s data could be written as events to a message queue and ingested using streaming platforms such as Amazon Kinesis or Apache Kafka
- But batch processing is still a prevalent requirement in downstream processing

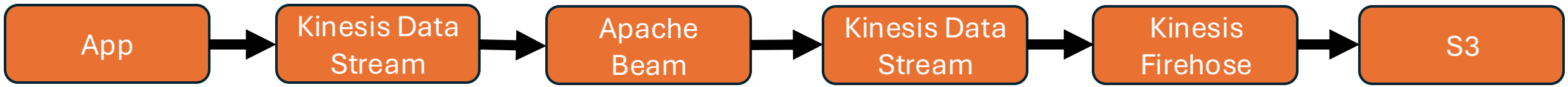


Synchronous vs Asynchronous Ingestion

- Synchronous: the source, ingestion, and the destination have complex dependencies and are tightly coupled.



- Asynchronous ingestion: dependencies can operate at the level of individual events



Serialization and Deserialization

- Serialization: encoding the data from a source and preparing data structures for transmission and intermediate storage
- Deserialization: opposite process
  - Ensure that your destination can deserialize the data

Throughput and Scalability

- Data throughput and system scalability become critical as your data volumes grow and requirements change
- Design systems to:
  - Scale effectively
  - Shrink flexibly
  - Handle bursty data ingestion using buffering
- Scenario: suppose your IoT sensor database storage goes down and when it comes back up, can it handle sudden influx of backlogged data?
- Best to leave throughput scaling to managed services:
  - More servers, shards or workers

Reliability and Durability

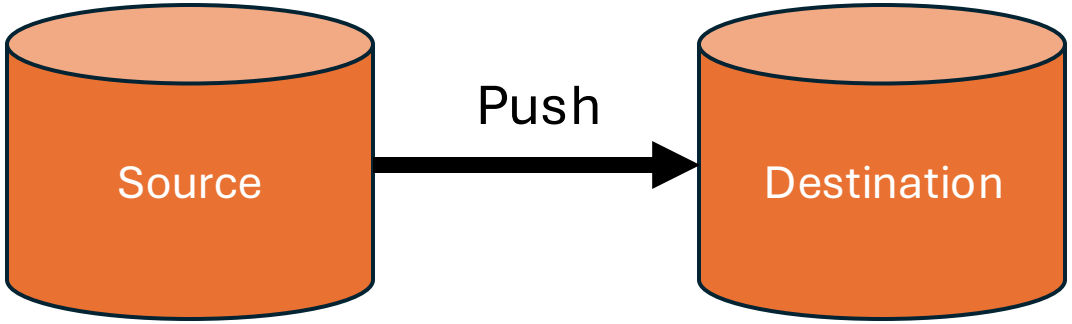
- Reliability: high uptime and proper failover for ingestion system
- Durability: data shouldn’t get lost or corrupted
- Build an appropriate level of redundancy:
  - Cost of data redundancy up vs cost of losing data

Payload

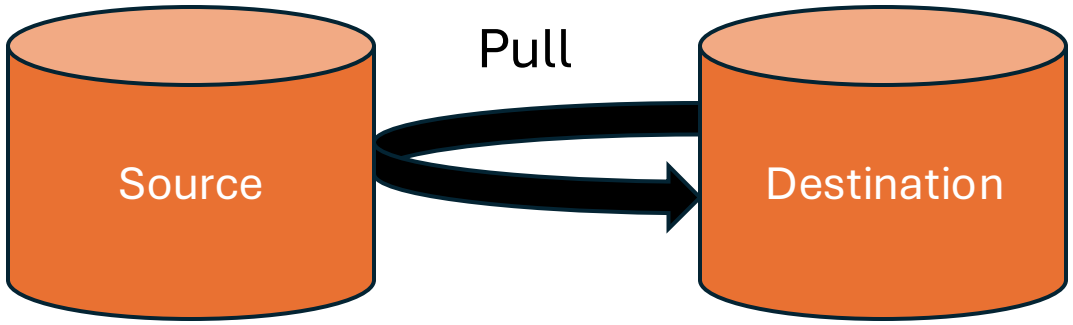
- Payload: characteristics of the ingested data:
  - Kind: data format ex: CSV, Parquet, JPG, PNG
  - Shape (dimensions) ex: Tabular, Semi-structured JSON, Unstructured text, Images, Uncompressed audio
  - Size: Number of bytes of payload → compression? Chunking and reassembly?
  - Schema and Data types
  - Meta data: data about data

Push vs Pull

- Push: source system sending data to a target

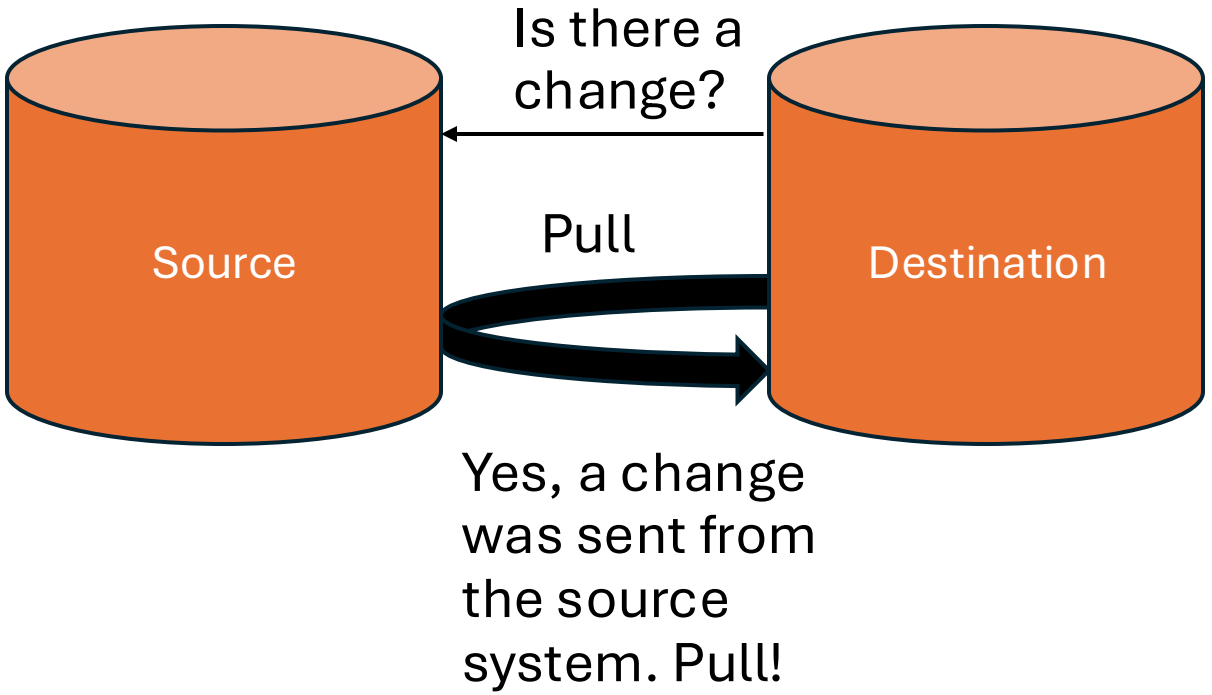


- Pull: target reading data from a source



Poll patterns

- Polling involves periodically checking a data source for any changes
- When changes are detected, the destination pulls the data



Upcoming tasks

Create Airbyte cloud account

Create Snowflake cloud account