Welcome

Advanced Data Engineering with Databricks

Learning Objectives

- **Design scalable ETL pipelines** using Databricks for batch and streaming data.
- **Optimize performance and costs** with partitioning, caching, and autoscaling.
- **Ensure data reliability** using Delta Lake ACID transactions and checkpointing.
- **Implement security and governance** with RBAC, encryption, and audit logging.
- **Orchestrate and monitor workflows** using Databricks Jobs, Airflow, and CI/CD.
- **Reduce infrastructure costs** with optimized cluster utilization and storage strategies.

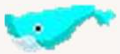
Modules

Welcome!

Atin Gupta

- Microsoft Certified Trainer
- Having 23+ years of experience
- Delivered mor than 150 corporate training

Welcome!

Let's get to know you 

- Name
- Role and team
- Length of experience with Spark and Databricks
- Motivation for attending

Architecting for the Lakehouse

Adopting the Lakehouse Architecture
Lakehouse Medallion Architecture
Streaming Design Patterns

Adopting the Lakehouse Architecture

Data Lake

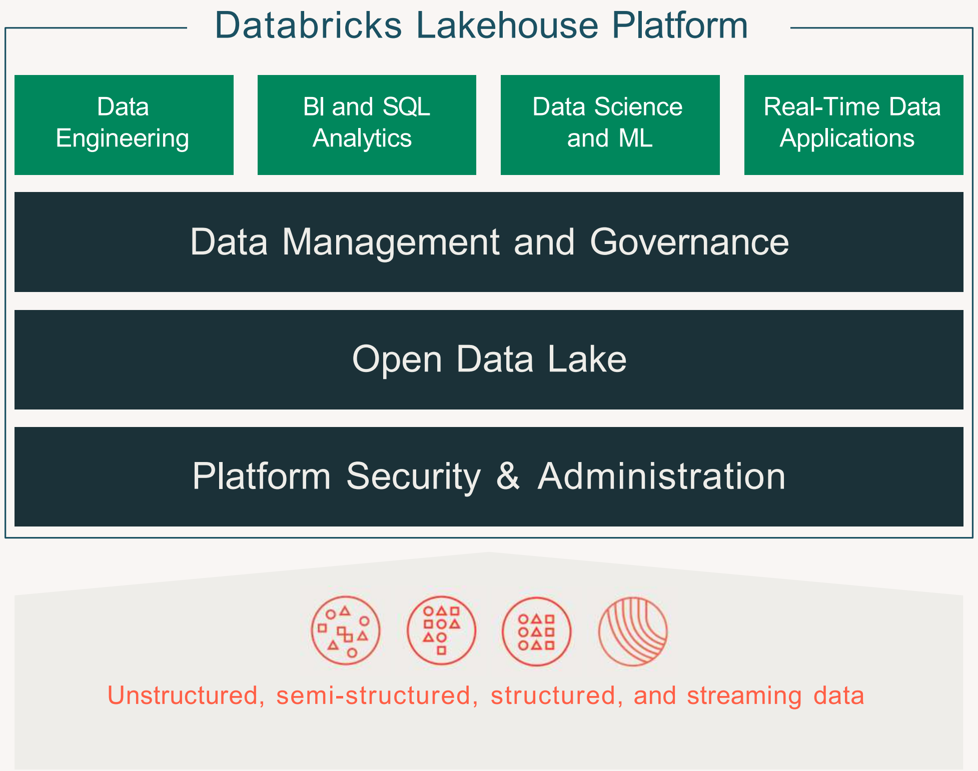
Lakehouse

One platform to unify all of your data, analytics, and AI workloads

Data Warehouse

The Databricks Lakehouse Platform

- Simple
- Open
- Collaborative



Data Lake

DELTA LAKE

An open approach to bringing **data management and governance** to data lakes

Better reliability with transactions

48x faster data processing with indexing

Data governance at scale with fine-grained access control lists

Data Warehouse

Delta Lake brings ACID to object storage

- Atomicity
- Consistency
- Isolation
- Durability

Delta Lake provides ACID guarantees scoped to tables

The Lakehouse Medallion Architecture

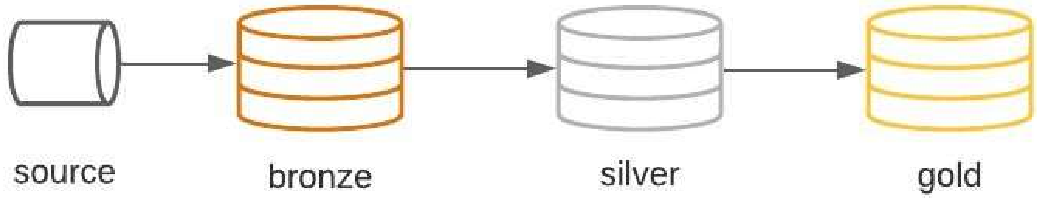
Multi-hop Pipeline

Source:
    Files or integrated systems

Bronze:
    Raw data and metadata

Silver:
    Validated data with atomic grain

Gold:
    Refined, aggregated data



Bronze Layer

Why is the Bronze Layer Important?

- Bronze layer replaces the traditional data lake
- Represents the full, unprocessed history of the data
- Captures the provenance (what, when, and from where) of data loaded into the lakehouse
- Data is stored efficiently using Delta Lake
- If downstream layers discover later they need to ingest more, they can come back to the Bronze source to obtain it.

Silver Layer

Why is the Silver Layer important?

- Easier to query than the non-curated Bronze “data lake”
  - Data is clean
  - Transactions have ACID guarantees
- Captures the full history of business action modeled
- Reduces data storage complexity, latency, and redundancy

Gold Layer

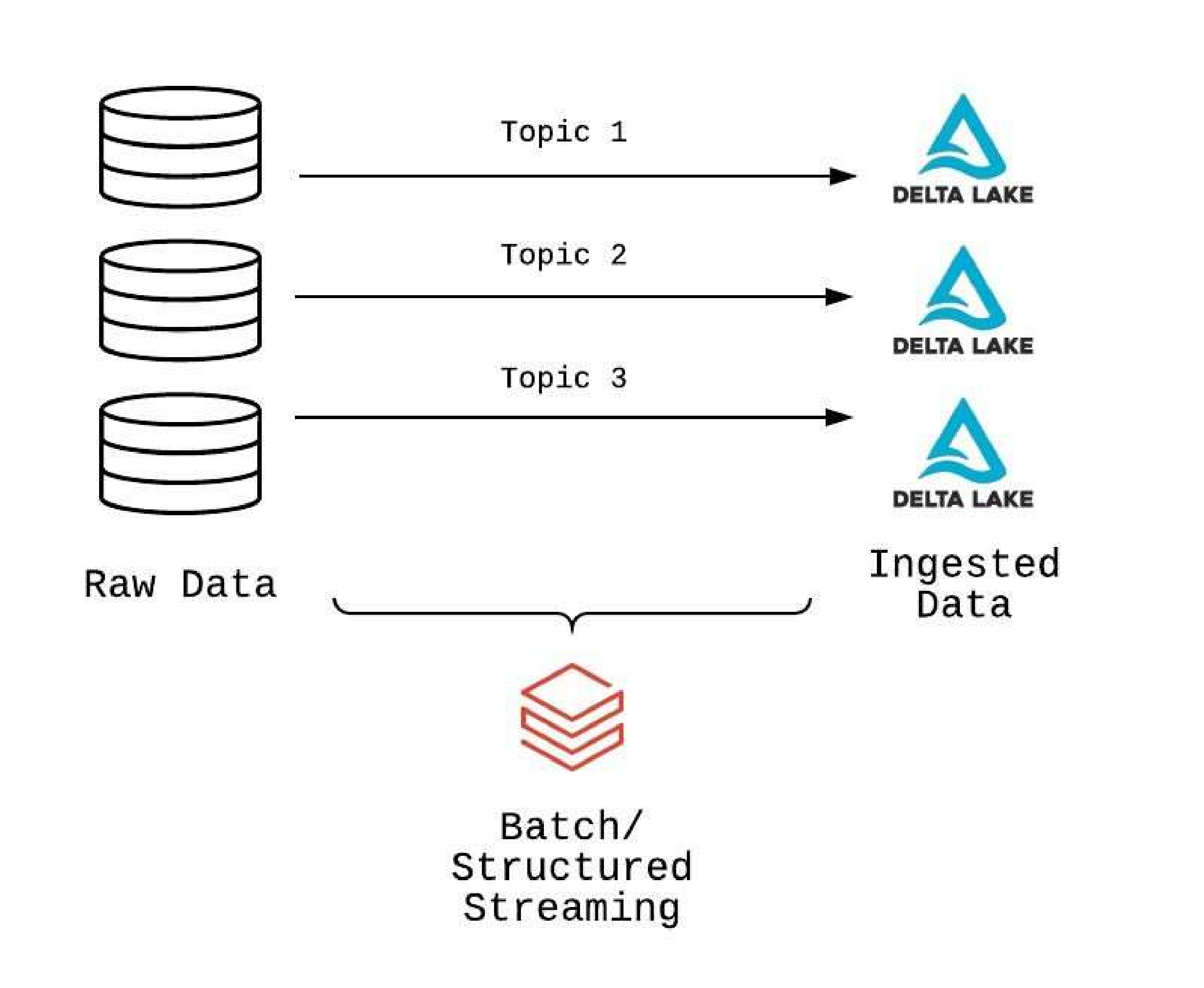
Why is the Gold Layer important?

- Powers ML applications, reporting, dashboards, ad hoc analytics
- Reduces costs associated with ad hoc queries on silver tables
- Allows fine grained permissions
- Reduces strain on production systems
- Shifts query updates to production workloads

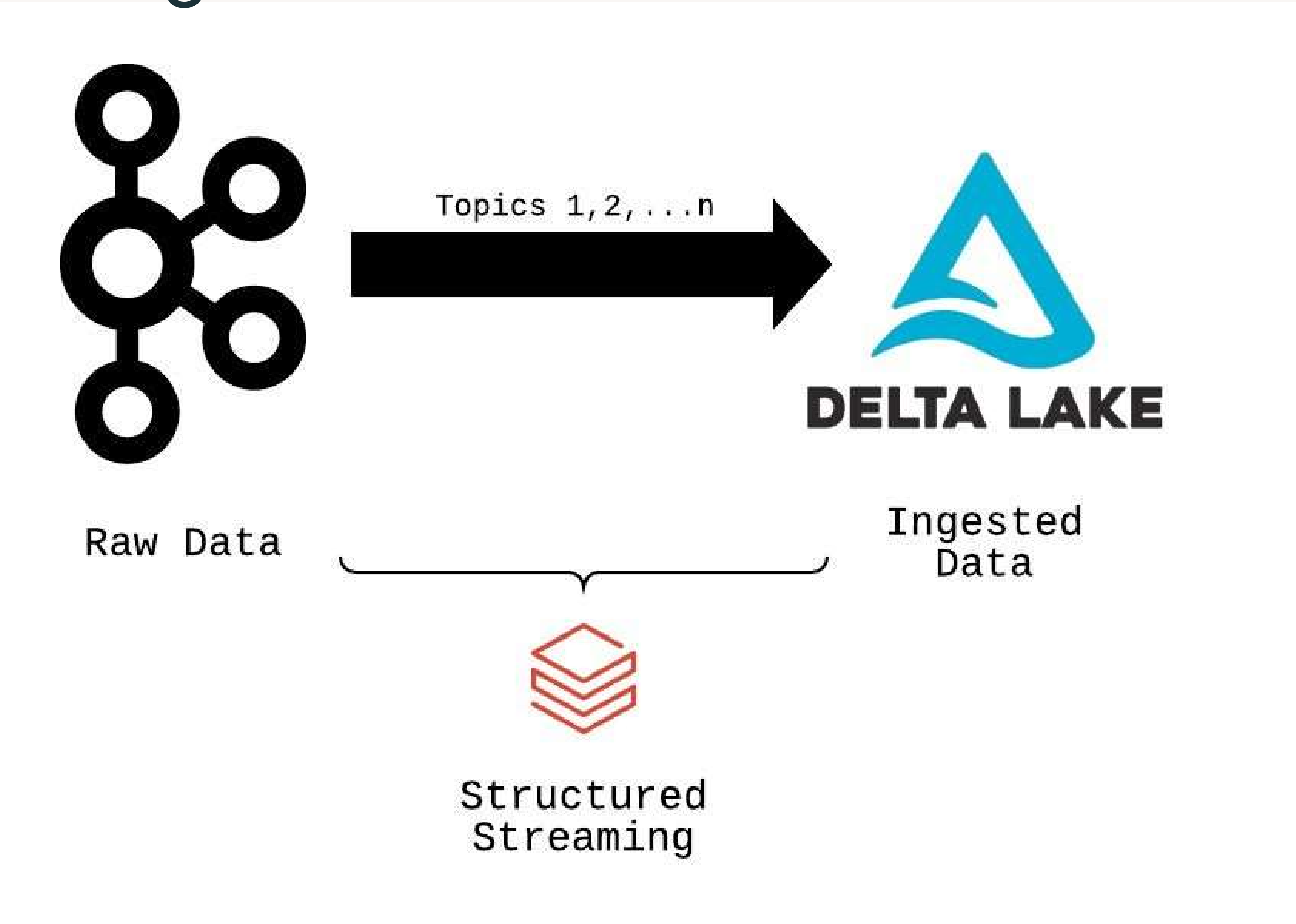
Bronze Ingestion Patterns

Bronze Ingestion Patterns
Auto Load to Multiplex Bronze
Streaming from Multiplex Bronze

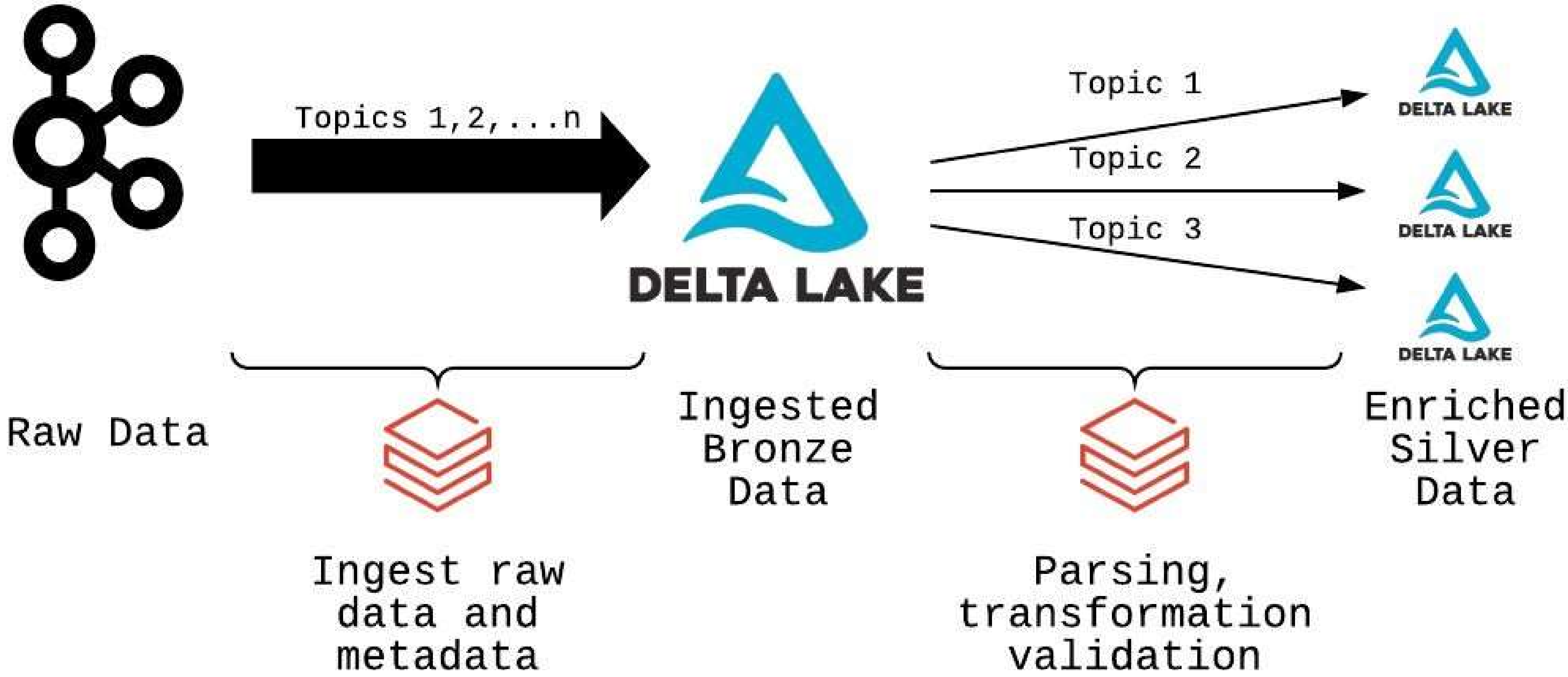
Singleplex Ingestion



Multiplex Ingestion



Delta Lake Bronze



Promoting to Silver

Streaming Deduplication
Quality Enforcement
Slowly Changing Dimensions
Streaming Joins and Statefulness

Silver Layer Objectives

- Validate data quality and schema
- Enrich and transform data
- Optimize data layout and storage for downstream queries
- Provide single source of truth for analytics

Schema Enforcement & Evolution

- Enforcement prevents bad records from entering table
  - Mismatch in type or field name
- Evolution allows new fields to be added
  - Useful when schema changes in production/new fields added to nested data
  - **Cannot** use evolution to remove fields
  - All previous records will show newly added field as Null
    - For previously written records, the underlying file isn't modified.
    - The additional field is simply defined in the metadata and dynamically read as null

Delta Lake Constraints

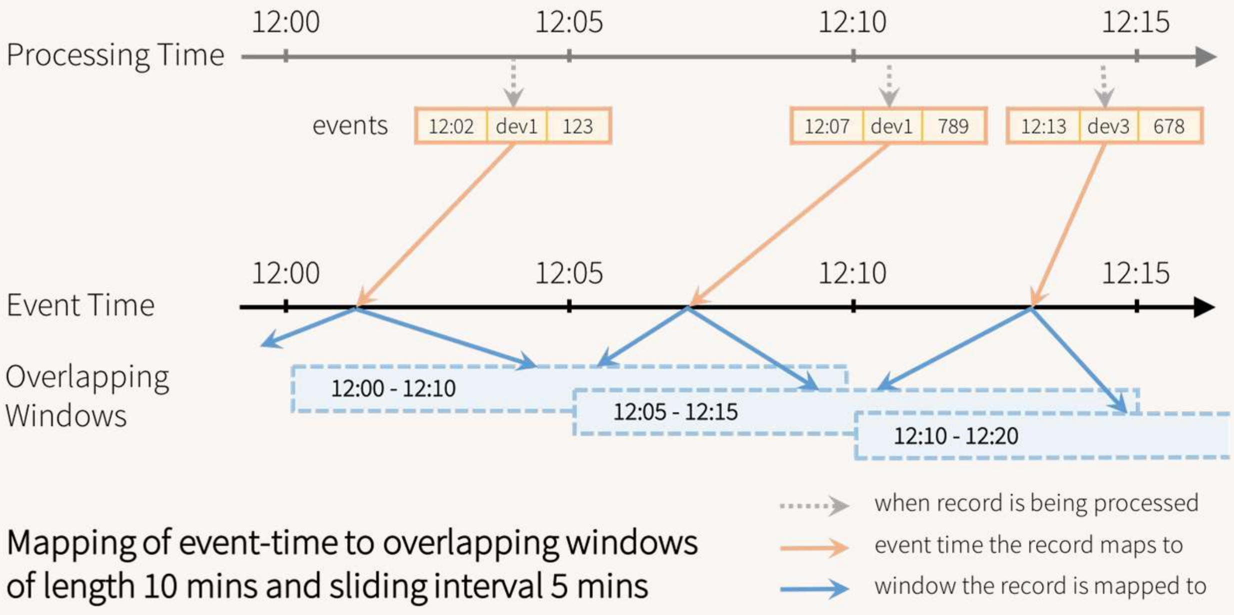
- Check `NOT NULL` or arbitrary boolean condition
- Throws exception on failure

```
ALTER TABLE tableName ADD CONSTRAINT constraintName
    CHECK heartRate >= 0;
```

Streaming Joins and Statefulness

The Components of a Stateful Stream

```
windowedDF =
  (eventsDF
   .groupBy(window("eventTime",
                   "10 minutes",
                   "5 minutes"))
   .count()
   .writeStream
   .trigger(processingTime="5 minutes")
   )
```



Mapping of event-time to overlapping windows of length 10 mins and sliding interval 5 mins

Output Modes

Mode	**When Stateful Results Materialize**
Append (default)	Only materialize after watermark + lateness passed
Complete	Materialize every trigger, outputs complete table
Update	Materialize every trigger, outputs only new values

Gold Query Layer

Making Data Available for Analytics
Stored Views
Materialized Gold Tables

What is the Query Layer?

- Stores refined datasets for use by data scientists
- Serves results for pre-computed ML models
- Contains enriched, aggregated views for use by analysts
- Star-schemas and data marts for BI queries
- Powers data-driven applications, dashboards, and reports

Also called the serving layer; gold tables exist at this level.

Storing Data Securely

PII & Regulatory Compliance
Storing PII Securely
Granting Privileged Access to PII

PII & Regulatory Compliance

Regulatory Compliance

- EU = GDPR (General Data Protection Regulation)
- US = CCPA (California Consumer Privacy Act)
- Simplified Compliance Requirements
  - Inform customers what personal information is collected
  - Delete, update, or export personal information as requested
  - Process request in a timely fashion (30 days)

How Lakehouse Simplifies Compliance

- Reduce copies of your PII
- Find personal information quickly
- Reliably change, delete, or export data
- Use transaction logs for auditing

Manage Access to PII

- Control access to storage locations with cloud permissions
- Limit human access to raw data
- Pseudonymize records on ingestion
- Use table ACLs to manage user permissions
- Configure dynamic views for data redaction
- Remove identifying details from demographic views

Pseudonymization

Pseudonymization

- Switches original data point with pseudonym for later re-identification
- Only authorized users will have access to keys/hash/table for re-identification
- Protects datasets on record level for machine learning
- A pseudonym is still considered to be personal data according to the GDPR

Anonymization

Anonymization

- Protects entire tables, databases or entire data catalogues mostly for Business Intelligence
- Personal data is irreversibly altered in such a way that a data subject can no longer be identified directly or indirectly
- Usually a combination of more than one technique used in real-world scenarios

Data Suppression

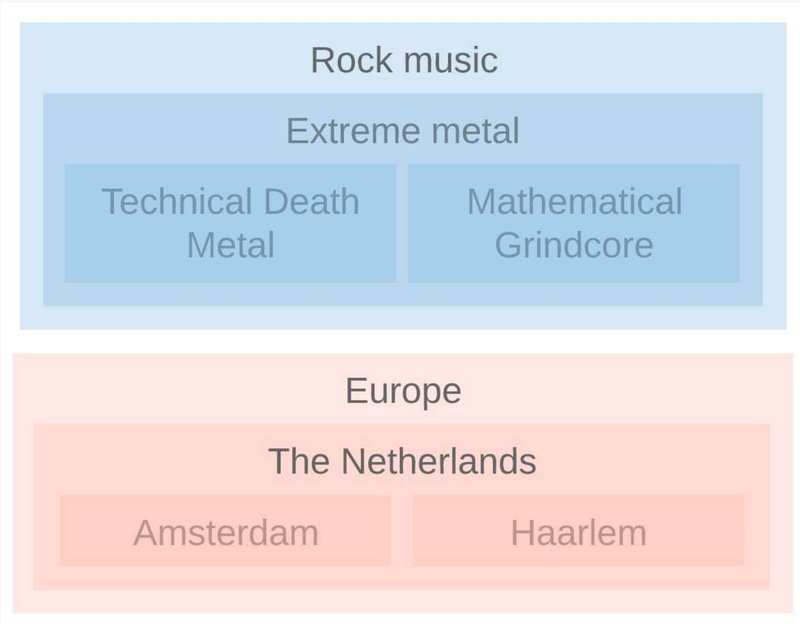
- Exclude columns with PII from views
- Remove rows where demographic groups are too small
- Use dynamic access controls to provide conditional access to full data

Generalization

- Categorical generalization
- Binning
- Truncating IP addresses
- Rounding

Categorical Generalization

- Removes precision from data
- Move from specific categories to more general
- Retain level of specificity that still provides insight without revealing identity



Managing ACLs for the Enterprise Lakehouse

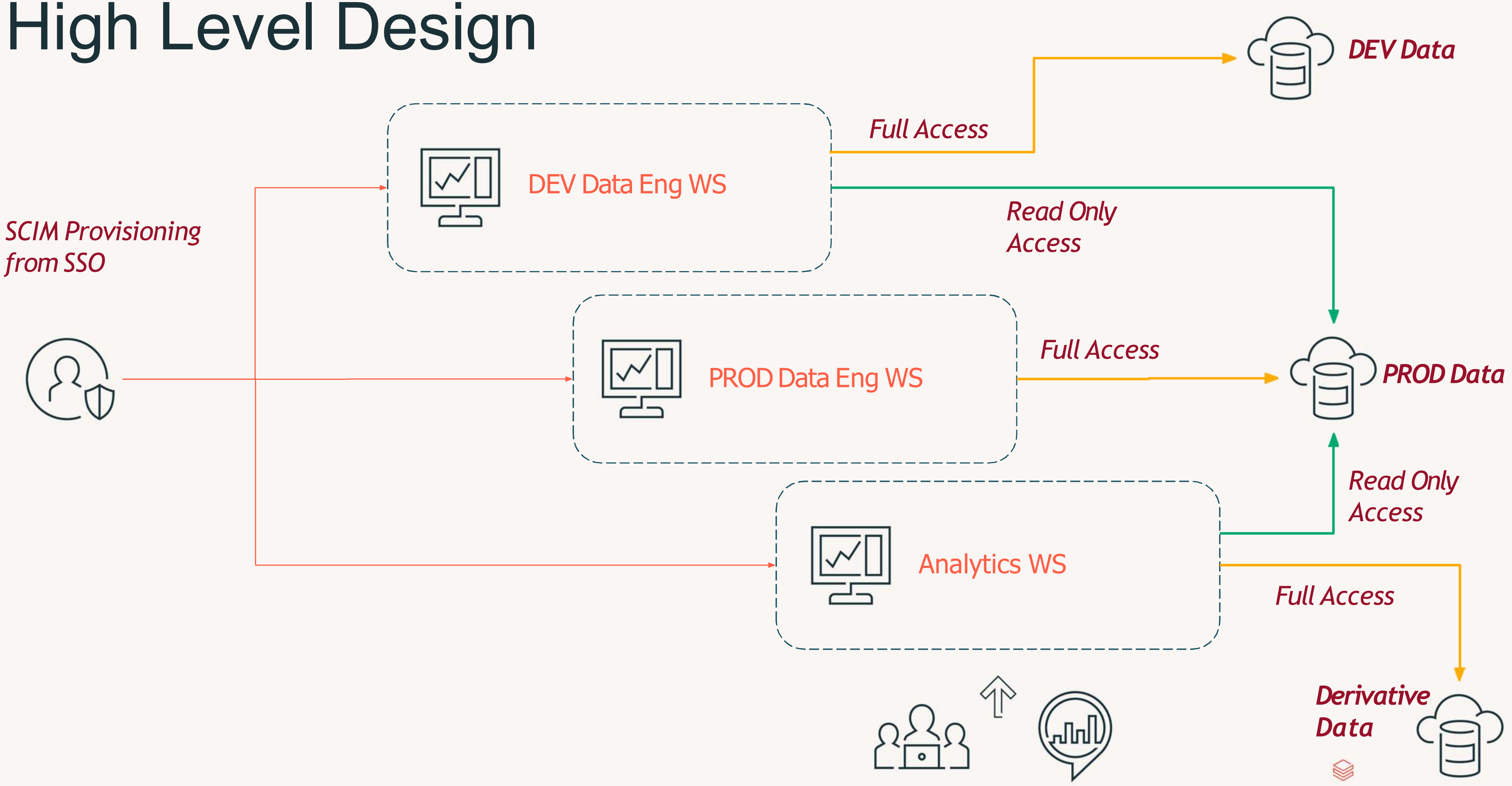
The Goal

Provide access to valuable data to users across the company in a secure manner.

Ensure users are only able to access the data they're entitled to.

Detect whether any data has been altered or manipulated.

High Level Design



Grant Access to Production Datasets

Assumptions

- End-users need read-only access
- Datasets organized by database

```
GRANT USAGE, SELECT, READ_METADATA ON DATABASE hr TO `HR`
GRANT USAGE ON DATABASE hr TO  `HR`;
```

Alternative, grant access on specific tables:

```
GRANT SELECT, READ_METADATA ON TABLE employees TO `HR`;
GRANT SELECT, READ_METADATA ON TABLE addresses TO `HR`;
```

Dynamic Views on Databricks

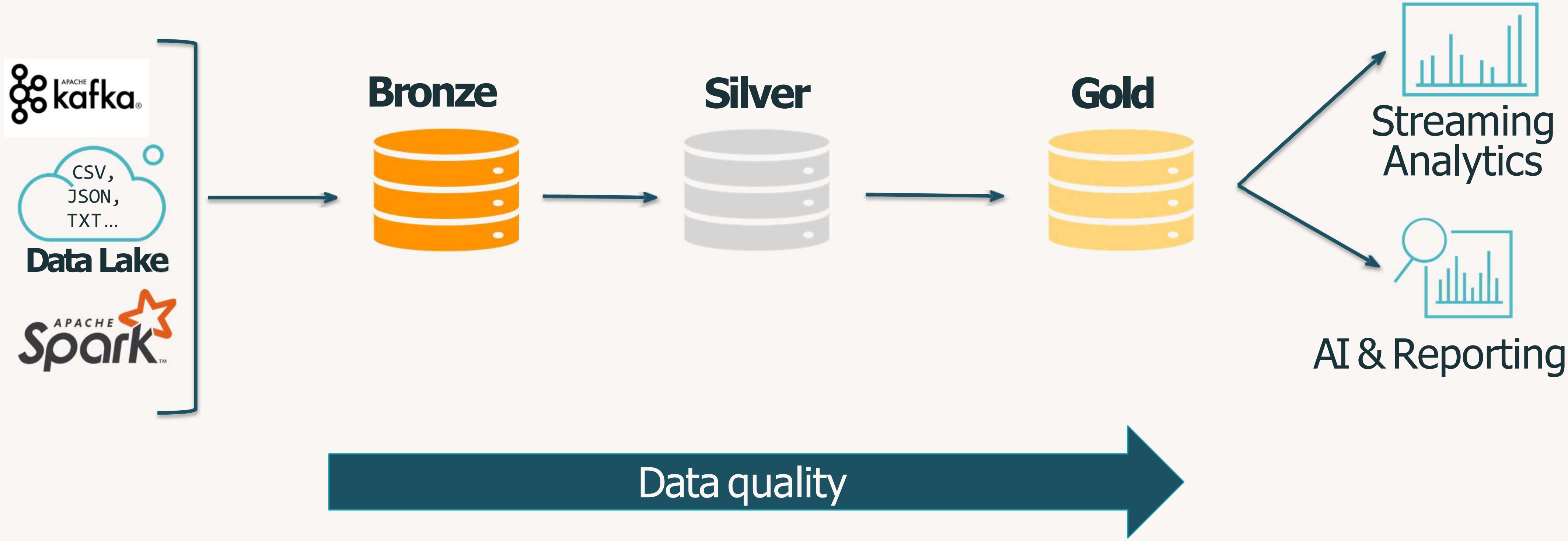
- Need to redact fields based on user's identity
- Do not give access to underlying table, only view
- Uses existing group membership to filter rows or columns

Propagating Updates and Deletes

Processing Records from Change Data Feed
Deleting Data in the Lakehouse

Propagating Changes with Delta Change Data Feed

Multi-Hop in the Lakehouse



What Delta Change Data Feed Does for You

Improve ETL pipelines

Process less data during ETL to increase efficiency of your pipelines

Unify batch and streaming

Common change format for batch and streaming updates, appends, and deletes

BI on your data lake

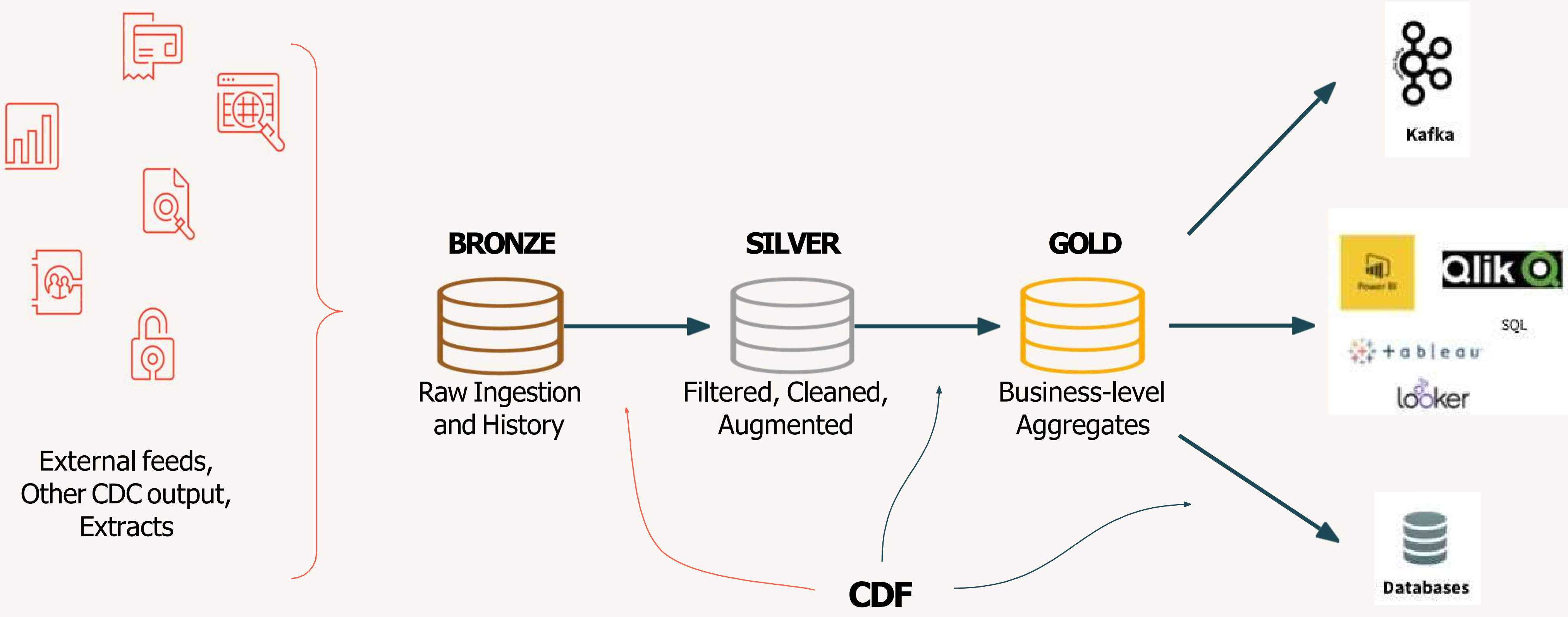
Incrementally update the data supporting your BI tool of choice

Meet regulatory needs

Full history available of changes made to the data, including deleted information

Delta Change Data Feed

Where Delta Change Data Feed Applies



How Does Delta Change Data Feed Work?

Original Table (v1)

	PK	B
	A1	B1
	A2	B2
	A3	B3

Change data (Merged as v2)

	PK	B
	A2	Z2
	A3	B3
	A4	B4

Change Data Feed Output

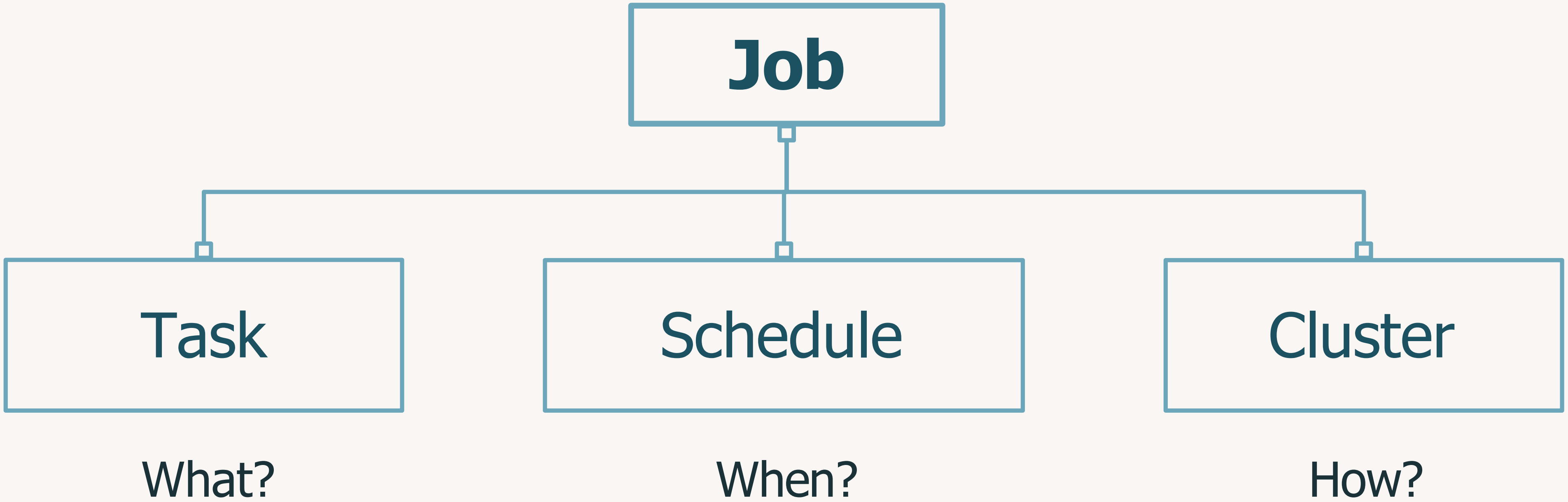
PK	B	Change Type	Time	Version
A2	B2	Preimage	12:00:00	2
A2	Z2	Postimage	12:00:00	2
A3	B3	Delete	12:00:00	2
A4	B4	Insert	12:00:00	2

A1 record did not receive an update or delete. So it will not be output by CDF.

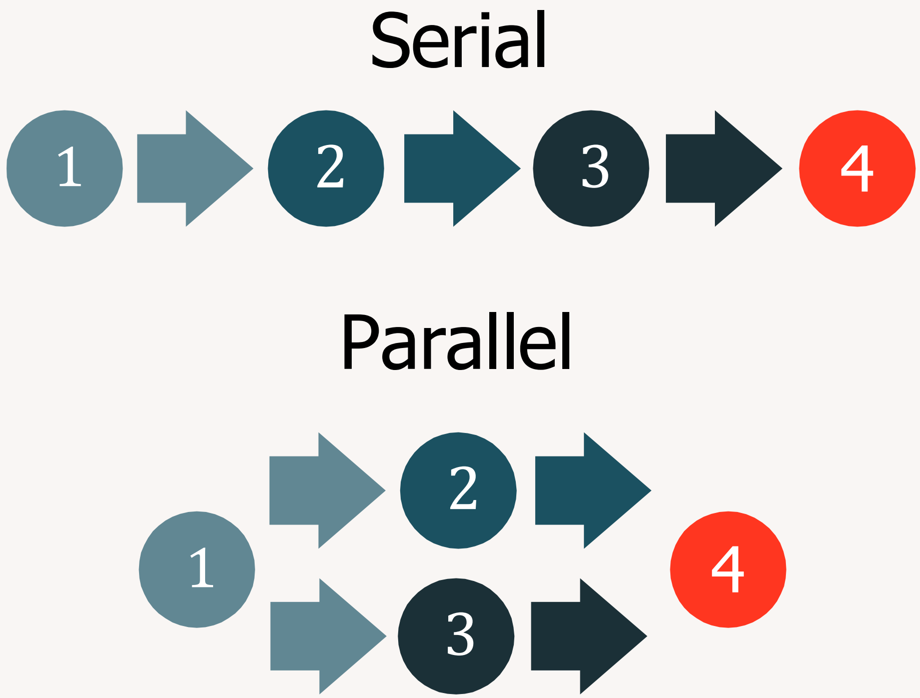
Orchestration and Scheduling

Multi-Task Jobs
Promoting Code with Repos
CLI and REST API
Deploying Workloads

What is a Job?

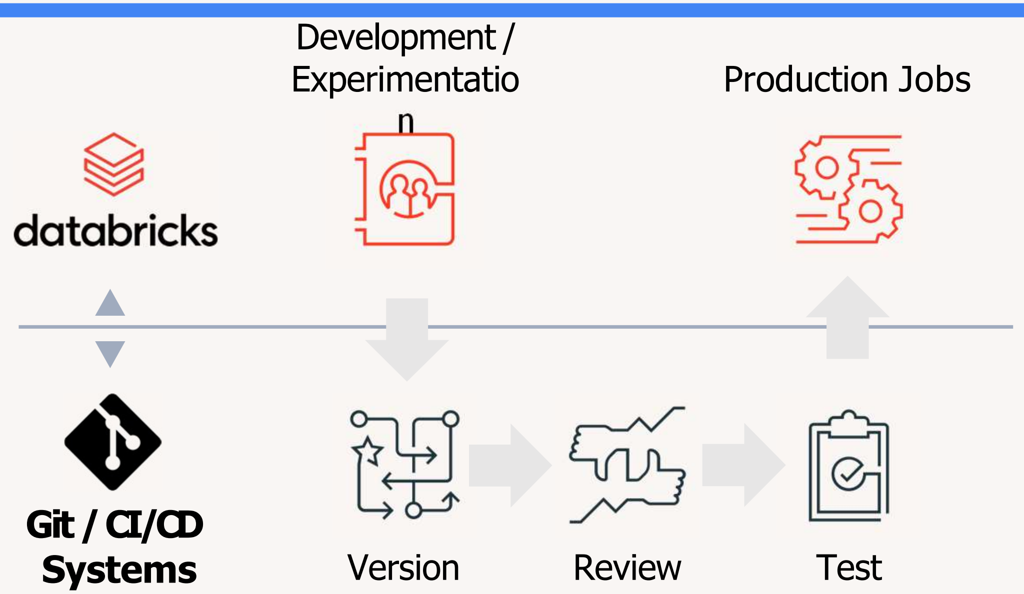


Orchestration with Multi-Task Jobs



Promoting Code with Databricks Repos

CI/CD Integration



Supported Git Providers

Azure DevOps GitHub GitLab Bitbucket