DATA+AI SUMMIT 2022

Near Real Time Event Analytics

Event Streaming, Delta Live Tables, and Delta Sharing

Christina Taylor
Senior Engineer, Special Projects @ Carvana

ORGANIZED BY databricks

Background

Microservices Architecture

A collection of loosely coupled application instances

Benefits

- Develop and test autonomously
- Scale independently
- Organize by business capability
- Separate code base
- Distributed CICD
- Data isolation

Criticism

- Complexity
- Network latency
- Information barriers
- Service boundaries
- Data integrity
- Data aggregation

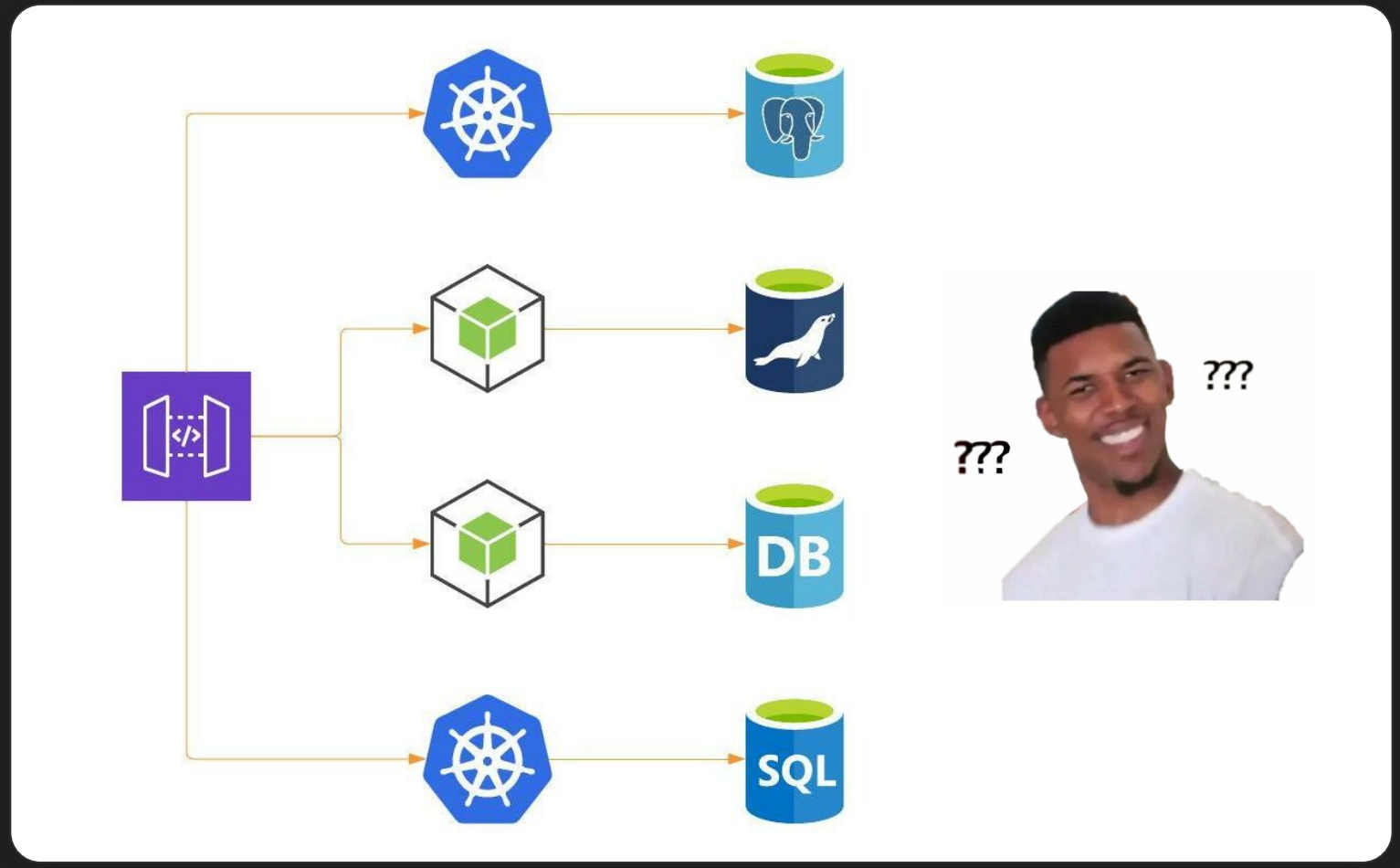
Conflicting Priorities

The challenges of application and data teams

Building Microservices

"Remember when we talked about the core principles behind good microservices? Strong cohesion and loose coupling - with database integration, we lose both things."

– Sam Newman



Change Data Capture

CDC

Delivers change data in real time to downstream processes

AWS Database Migration Service

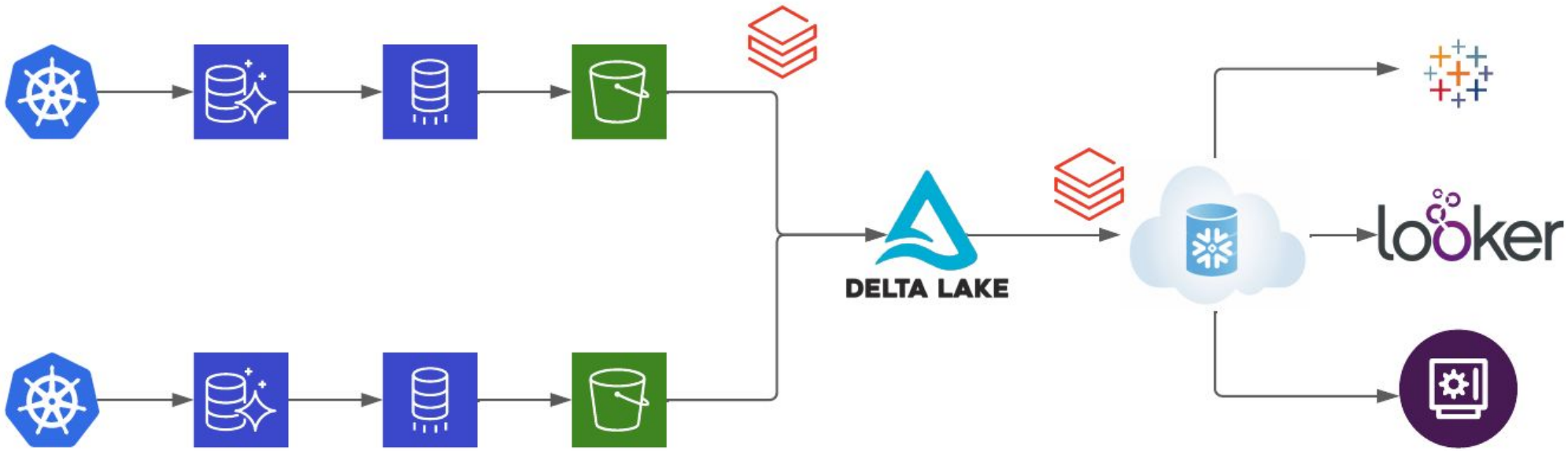
- One time snapshot
- Ongoing replication
- Supports a wide range of commercial and open source databases
- Supports both db and S3 targets
- Captures most DDL changes

Debezium

- Open source distributed platform
- Log based change data capture
- Kafka Connect/Debezium server
- Supports a variety of sources/targets
- Customization: filters, masking, and message transformation

System Architecture

AWS DMS + Databricks Autoloader for CDC Ingestion



Insights

When should we (not) use such a CDC system?

Scale

Works very well with big data

Scaling up to Terabytes

Stability

Eventual consistent

Works very well with stable systems where there are few DDL changes

Deployment

Automation of extra-connection attributes in Terraform is an art

Unsupported DDL and transaction blocks can be a mystery

Evolution

Schema change is immediately reflected in Delta Lake

Business logic is lost between service layer and data layer

Event Streaming

Event streams are sequences of business activities ordered by time. Data streaming consists of applications publishing and consuming events. Consumer programs aggregate, filter and enrich the information in near real time.

CloudEvents is a specification that describes event data in a common way.

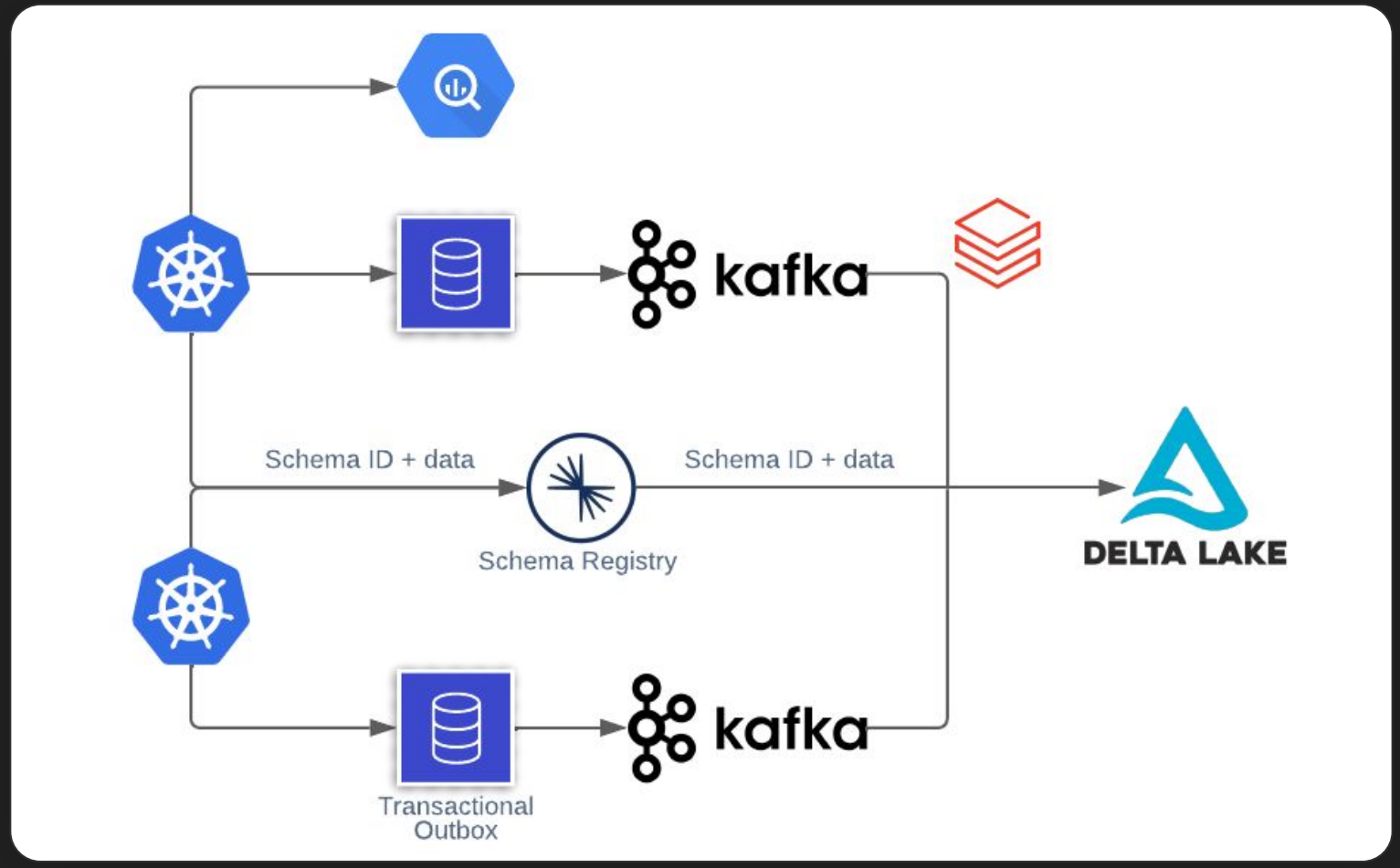
Header	Description	Example
ce-id	Unique identifier of the event	abcd-1234-5678
ce-source	Identifier of the source of the event	//company.api.com/1234/topics/order
ce-specversion	Specification version for this event	1.0
ce-type	CloudEvent type constant	google.cloud.pubsub.topic.v1.orderPlaced
ce-time	Time when the message was sent	2022-01-01T10:11:22.333Z
... further attributes ...		
ce-schemaid	Payload schema id (as stored in schema registry)	100001
ce-schemaversion	Payload schema version	0, 1, latest

System Architecture

Establish contract between data producers and consumers

Components

- Transactional Outbox
- “Scanner”
- Message queue
- Producers
- Consumers
- Schema Registry



Schema Registry Service

A RESTFUL interface for storing and receiving schemas

```
{
  "name": "checkout",
  "type": "record",
  "fields": [
    {
      "name": "customer_name",
      "type": "string",
      "aliases": ["name", “user_name”],
      "doc": "the name customer chose at checkout"
    },...
  ]
}
```

Compatibility

- Forward
- Forward transitive
- Backward
- Backward transitive
- Full
- Full transitive
- None

Spark Streaming

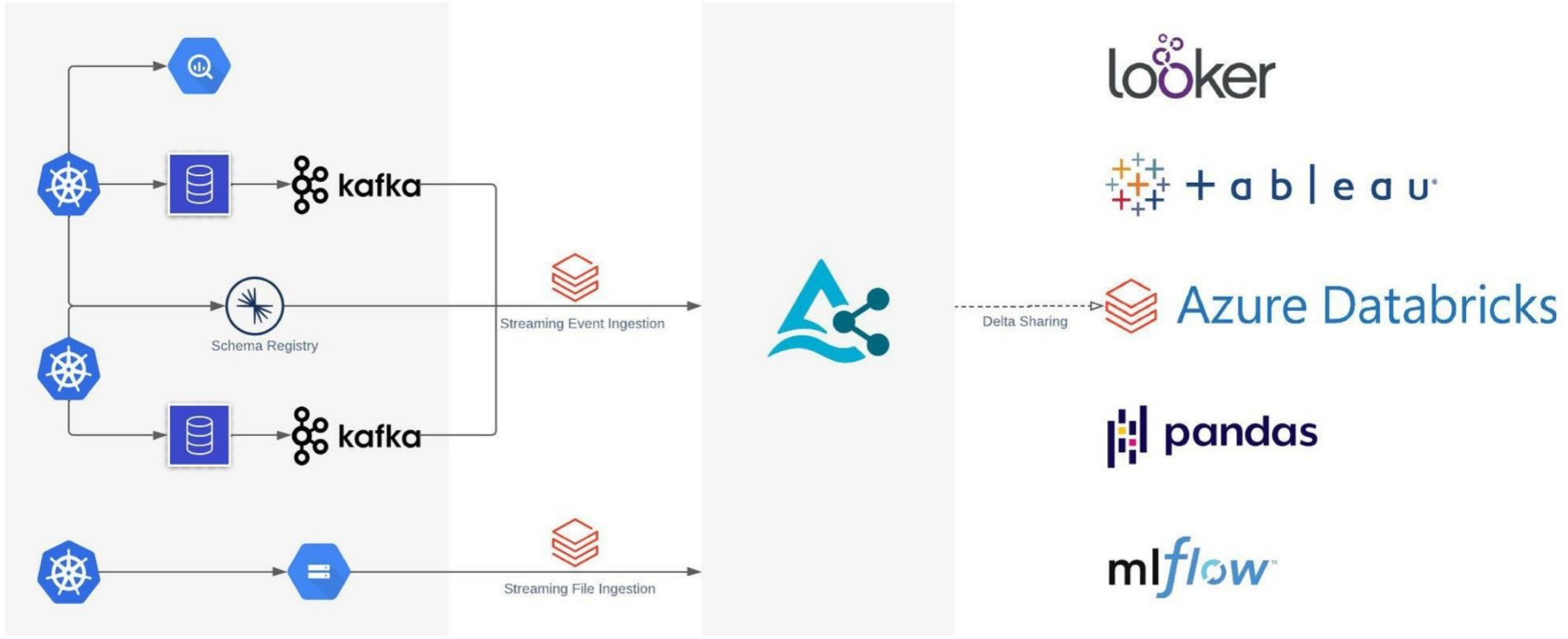
```
1   // Deserialize Avro messages
2   import za.co.absa.abris.config.{AbrisConfig, FromAvroConfig}
3   val commonRegistryConfig = ...
4   val valueRegistryConfig: Map[String, String] = commonRegistryConfig ++ Map(
5       "value.schema.naming.strategy" -> "topic.name"
6       "value.schema.id" -> "latest",
7       "value.schema.version" -> "latest"
8   )
9   val abrisConfig: FromAvroConfig =
10      AbrisConfig.fromConfluentAvro.downloadReaderSchemaByLatestVersion
11        .andTopicNameStrategy(topic)
12        .usingSchemaRegistry(valueRegistryConfig)
```

Spark Streaming

```
1   // Reformat Confluent Avro messages
2   def reshapeConfluentKafkaMessage(fromAvroConfig: FromAvroConfig)(df: DataFrame): DataFrame = {
3       df.withColumn("data", from_avro(col("value"), fromAvroConfig))
4           .select("data.*", "*")
5           .drop("data", "value")
6   }
7   // Extract schema information
8   kafkaTopic.df
9       .withColumn("headers_struct", map_from_entries(col("headers")))
10      .withColumn("schemaVersion",col("headers_struct.ce_schemaversion").cast("STRING"))
11      .withColumm("schemaVersion",col("headers_struct.ce_schemaversion").cast("STRING"))
12      ...
```

Streaming Data Platform Ecosystem

Deliver and share near real-time insights



Delta Live Tables

Live Tables Pipeline

Coordinates data flow between queries

```
CREATE INCREMENTAL LIVE TABLE bronze_data
TBLPROPERTIES ("quality" = "bronze") AS
SELECT … FROM cloud_files("s3://bronze_bucket/*", "json",
        map("cloudFiles.schemaEvolutionMode", "rescue"))
```

Raw ingestion

```
import dlt
@dlt.create_table(table_properties={“quality”:”silver”})
def silver_data():
    return spark.read.csv("s3://silver_bucket/*.csv", header=True)
```

Filtered, cleansed, augmented

```
CREATE LIVE TABLE gold_data (CONSTRAINT pk EXCEPT (id IS NOT NULL) … )
TBLPROPERTIES ("quality" = "gold") AS
    SELECT … FROM LIVE.silver_data
    JOIN … ON …
```

Business level, aggregated

DLT Objectives

Simplify delivery of quality data with speed

Democratization

Collaborate more effectively by mixing python and SQL

Always-on

Continuous execution without complex stream processing and recovery logic

Quality

Define expectations and conformance to business rules

Visibility

Easily deploy declarative ("what") ETL pipelines with data flow graph dashboard

Delta Sharing

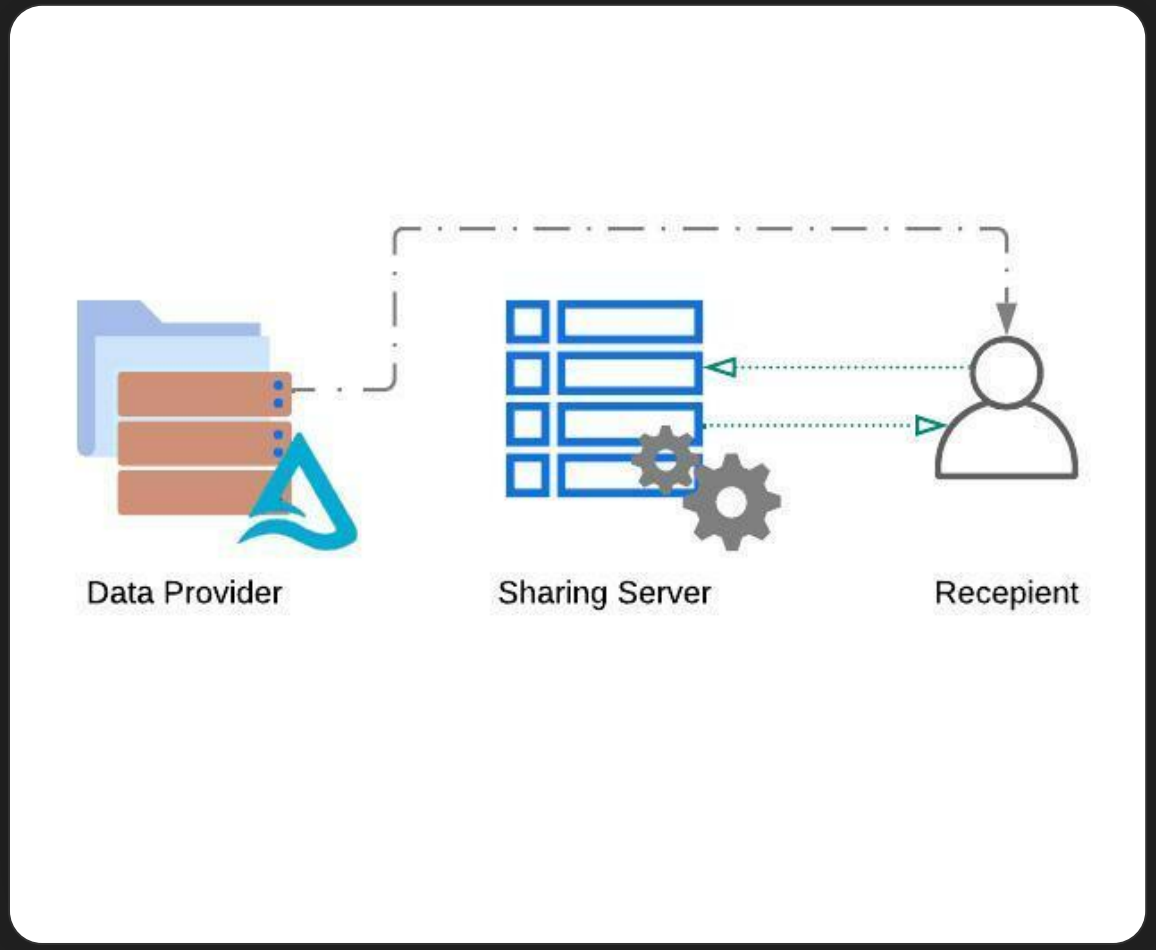
Delta Sharing Objectives

Securely share large amount of live data

Benefits

- Share live data without duplication
- Support a wide range of recipients
- Data security, audit and governance
- Scale to massive datasets

Recipients can always see a consistent view as opposed to external tables



DATA+AI
SUMMIT 2022

Thank you

Christina Taylor, Special Engineer of Projects
www.linkedin.com/in/ctaylor2