

# Apache Spark Based Reliable Data Ingestion in Datalake

Gagan Agrawal, Paytm

#SAISDev13

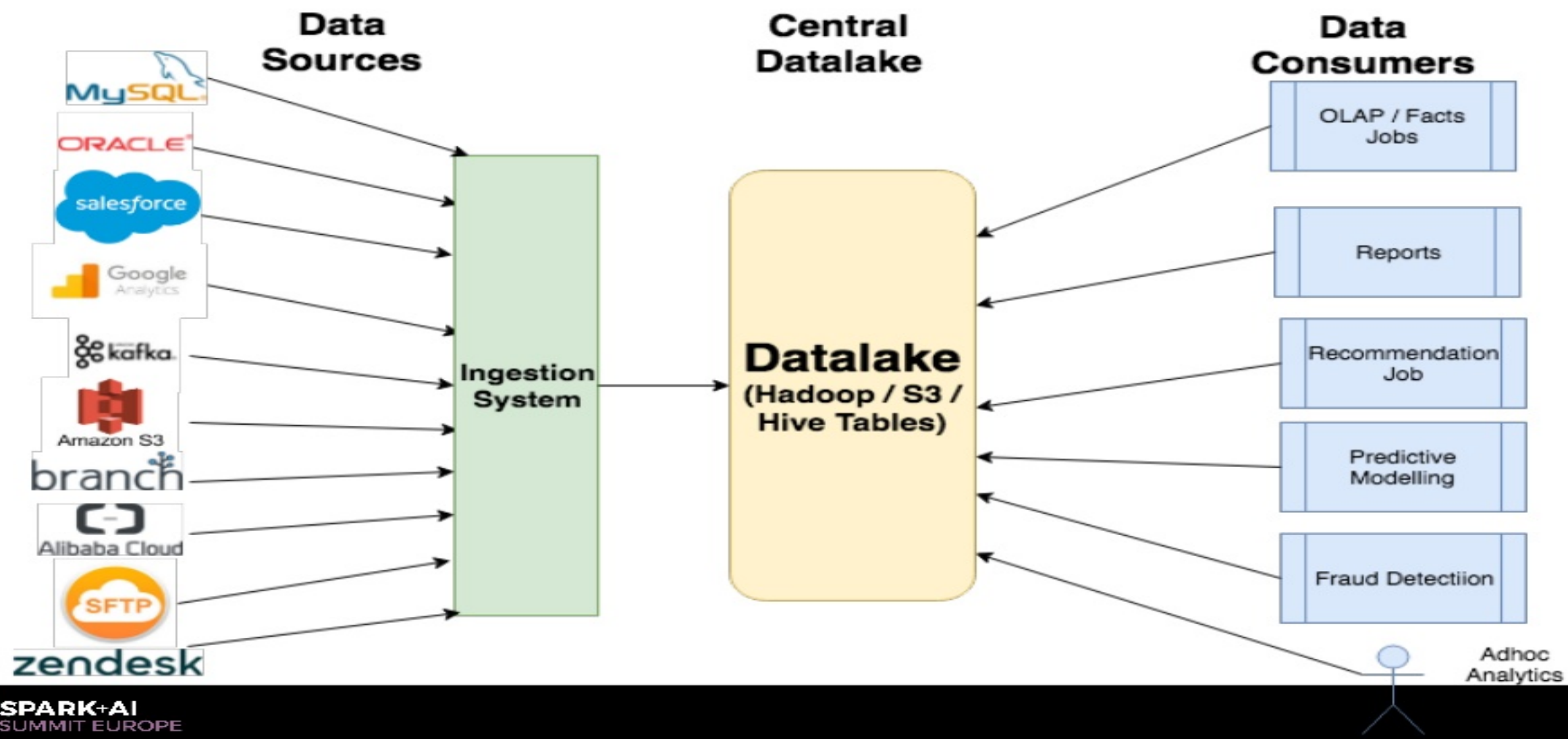
# Paytm - Who we are?

- Digital Payment Platform
- Wallet, Payment Banks, Payment Gateway, Paytm Money, e-Commerce, Recharges, Travel & Hotel booking etc.
- Fast growing company. New verticals/business added frequently
- 330M+ of registered users
- Generates multi TB data

# Agenda

- Ingestion Requirements
- Traditional Ingestion Mechanism
- Challenges
- Spark Based Reliable Ingestion Platform

# Ingestion Requirements



# Ingestion Types

- Full Ingestion
  - No checkpointing required
- Incremental Ingestion
  - Requires checkpointing
  - With or without compaction / de-duplication

Source	Configuration	State / Checkpoint
<b>MySQL / Oracle</b>	Host Port User Password	Timestamp e.g 2018-01-01 04:00:00
<b>Kafka</b>	Broker Host Topic	Offsets
<b>S3</b>	Bucket Access Key Secret Key	Last directory (Could be date based)
<b>SFTP</b>	Host Port User Password	Last directory (Could be date based)

# Traditional Ingestion Mechanism and it's Challenges



# Traditional Ingestion Mechanism

- Custom scripts for each source
- MySQL / Oracle -> Sqoop
- Salesforce -> Python script to ingest from SF API
- Kafka -> Kafka Connect
- S3 -> S3 Client API
- Cron based scheduler

# Challenges

- Variety of sources with their own source code
- More code = Higher chances of bug
- Custom configuration mechanism
- Custom state/checkpoint management
- Custom Transformation Mechanism
- Failure Handling / Retry / Backoff
- Monitoring
- Sanity / Data Correctness Checks

## More Challenges...

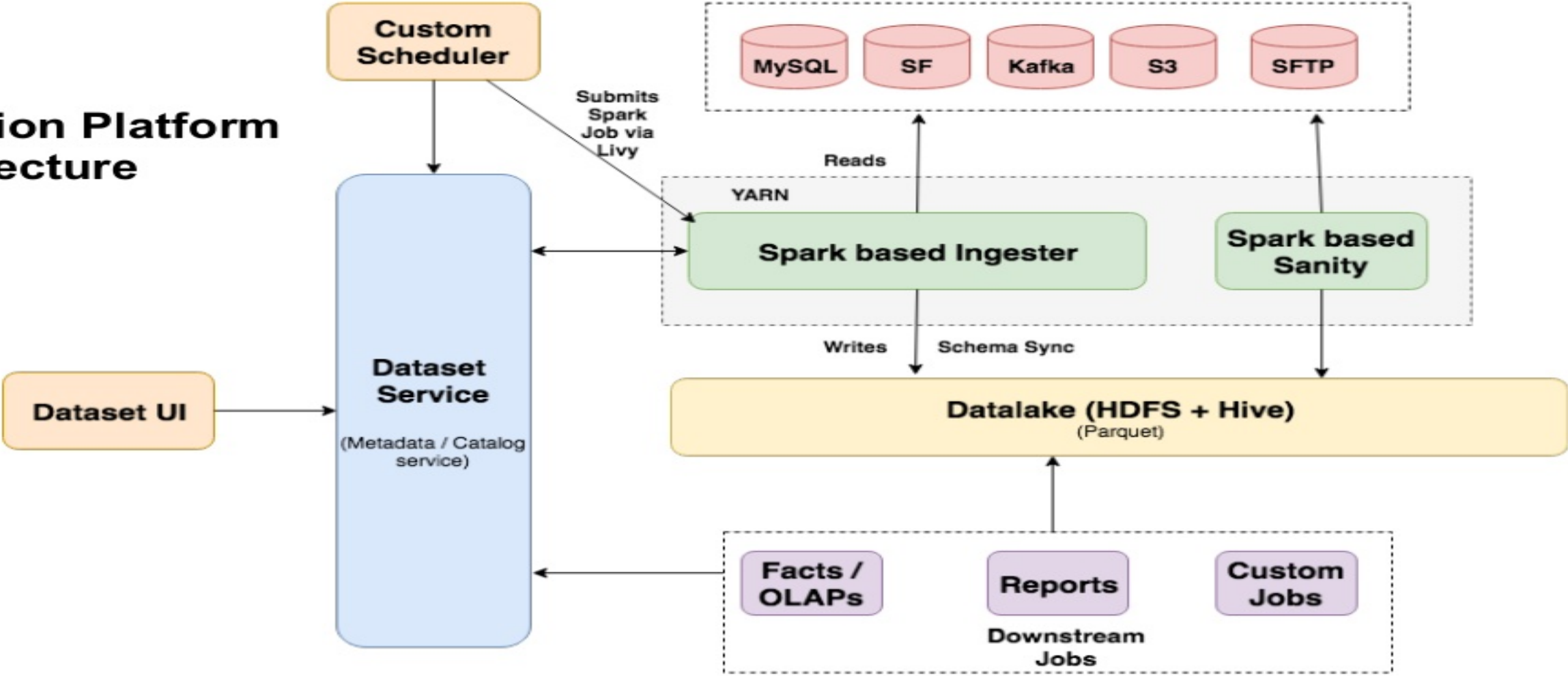
- Schema change at source not communicated
  - New Column Added
  - Datatype Change int -> bigint
  - Column Deleted
- Inconsistent Reads while Writes in progress
- Backfill / Re-Ingestion

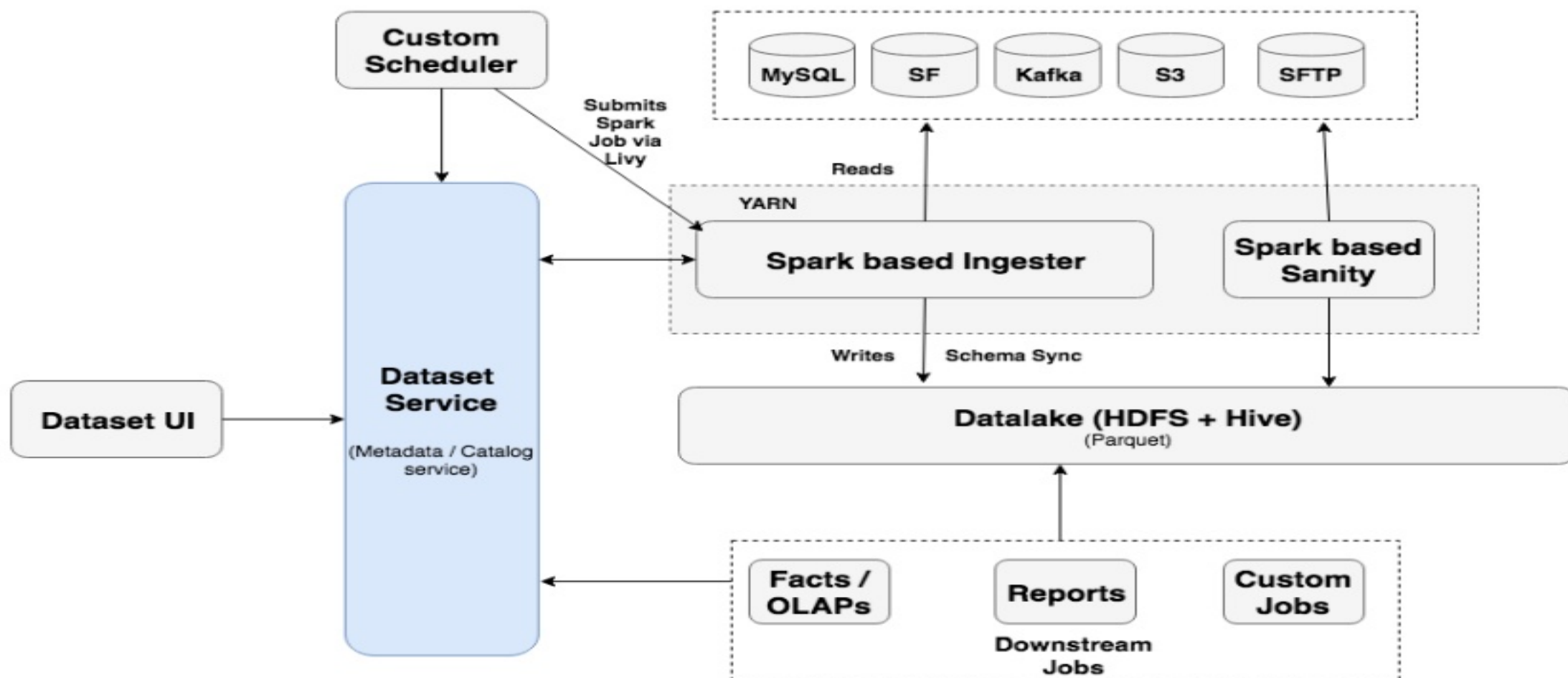
# Spark Based Ingestion Platform

# Spark Dataframe Abstraction

- Data Source API to connect to various sources
- Unified way of adding transformations
- Leveraging Spark Dataframe schema for auto schema change detection
- Unified way of finding max Record time / Processing time for downstream job dependency management

# Ingestion Platform Architecture





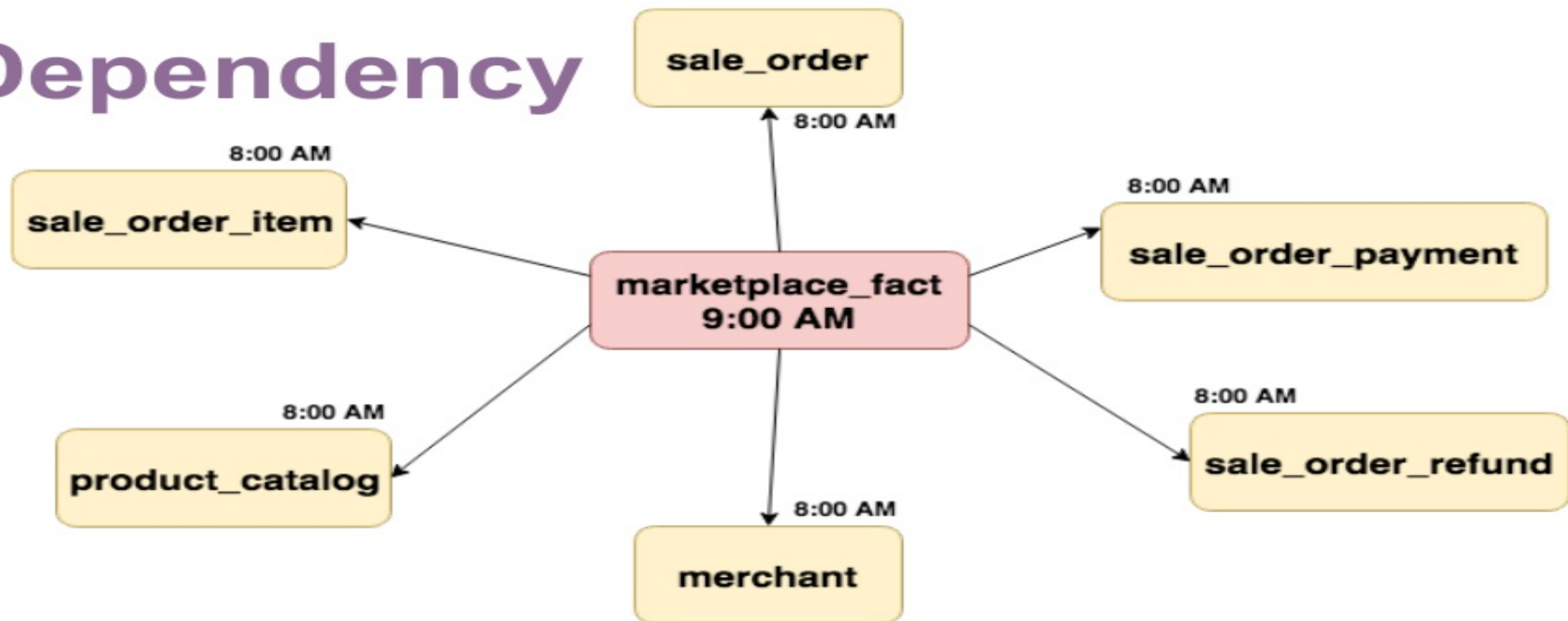


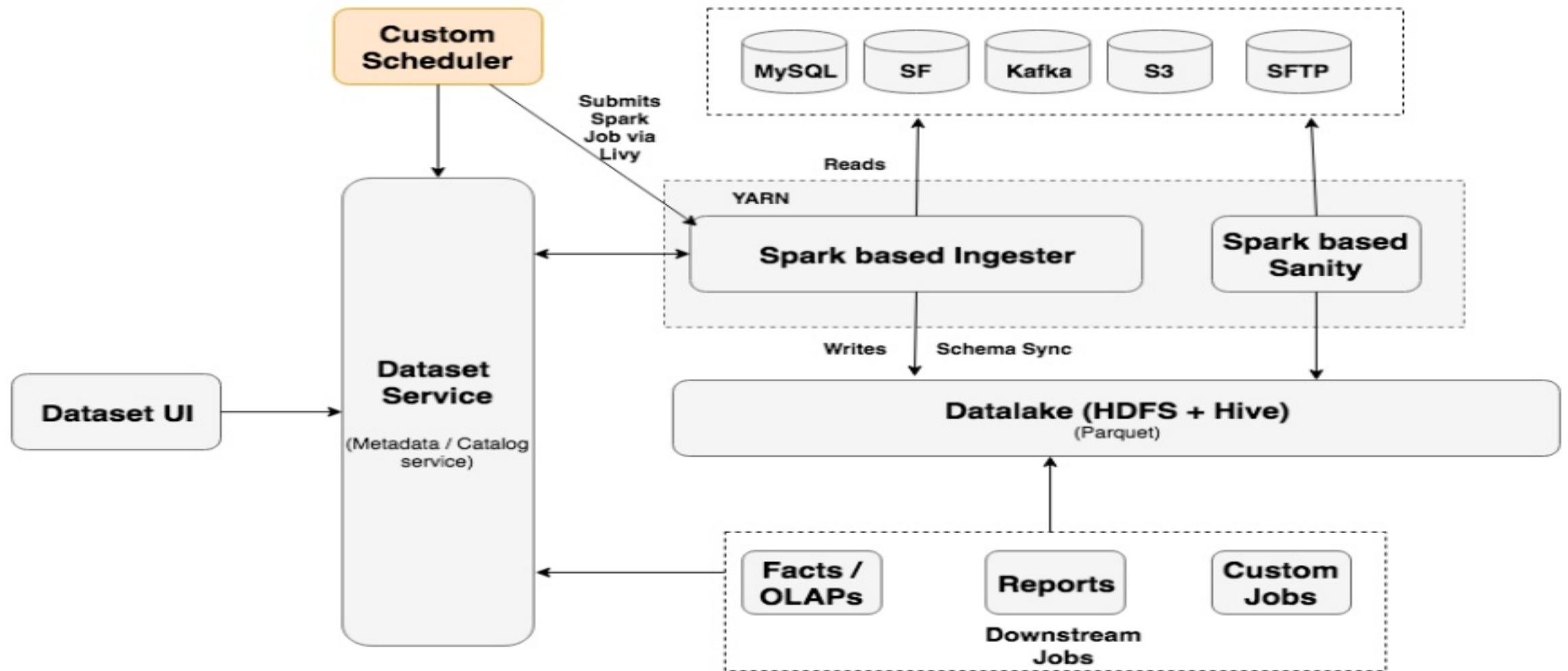
# Dataset Service

- Central Dataset Management Service
- Provides API to connect to source and fetch source details (available table, schema etc) for easy on-boarding
- Manages metadata like
  - State Checkpointing
  - Max Record / Processing time -> Till what time data is available
- Provides Dependency Management API
- Activate / Deactivate
- History Management (Schema changes, config changes etc.)
- Register Transformations



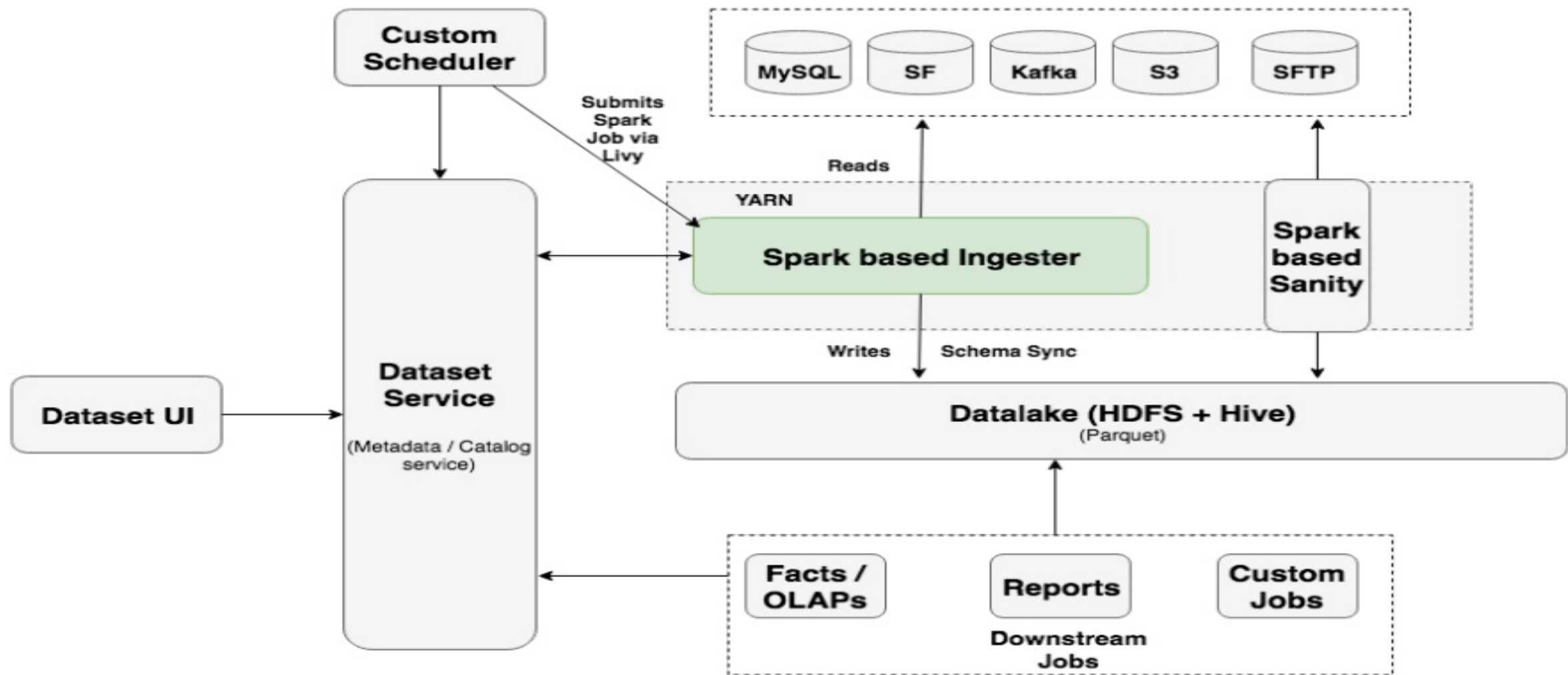
# Dependency



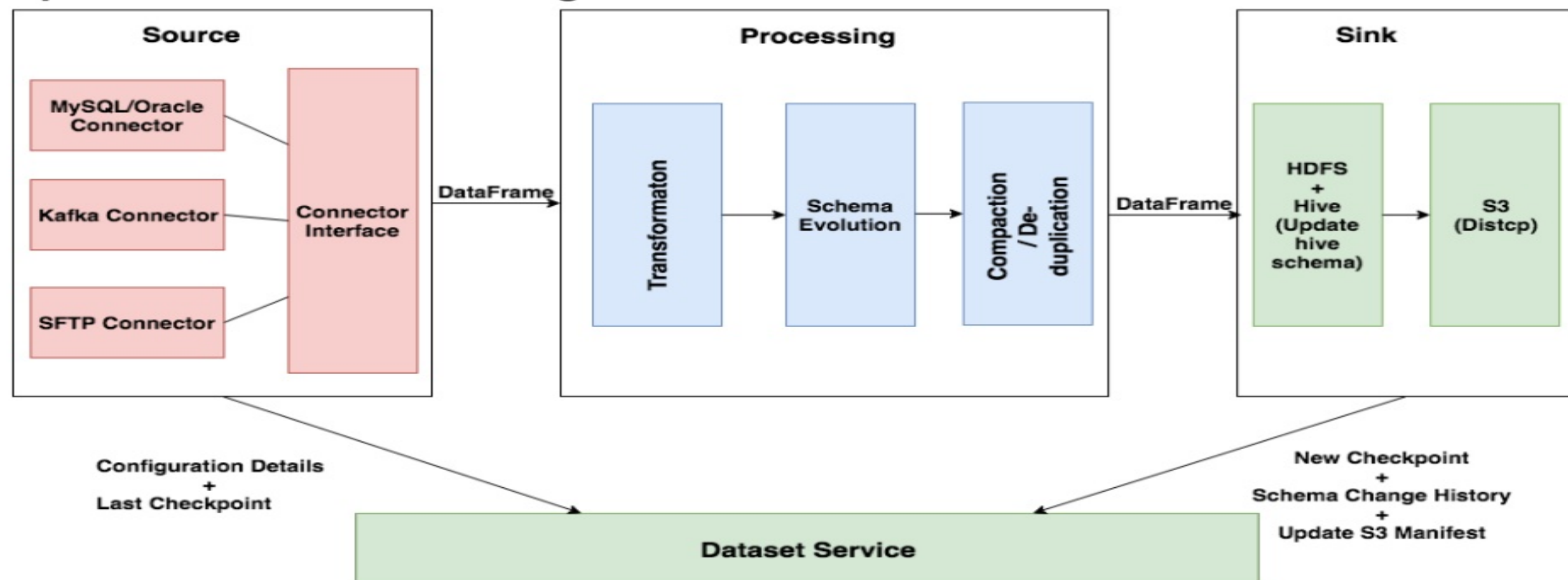


# Custom Scheduler

- SLO based scheduling
- High Priority Scheduling for Critical Reporting Datasets
- Max Connections based scheduling
- Submits Spark Job via Livy Rest API
- Retries / Backoff



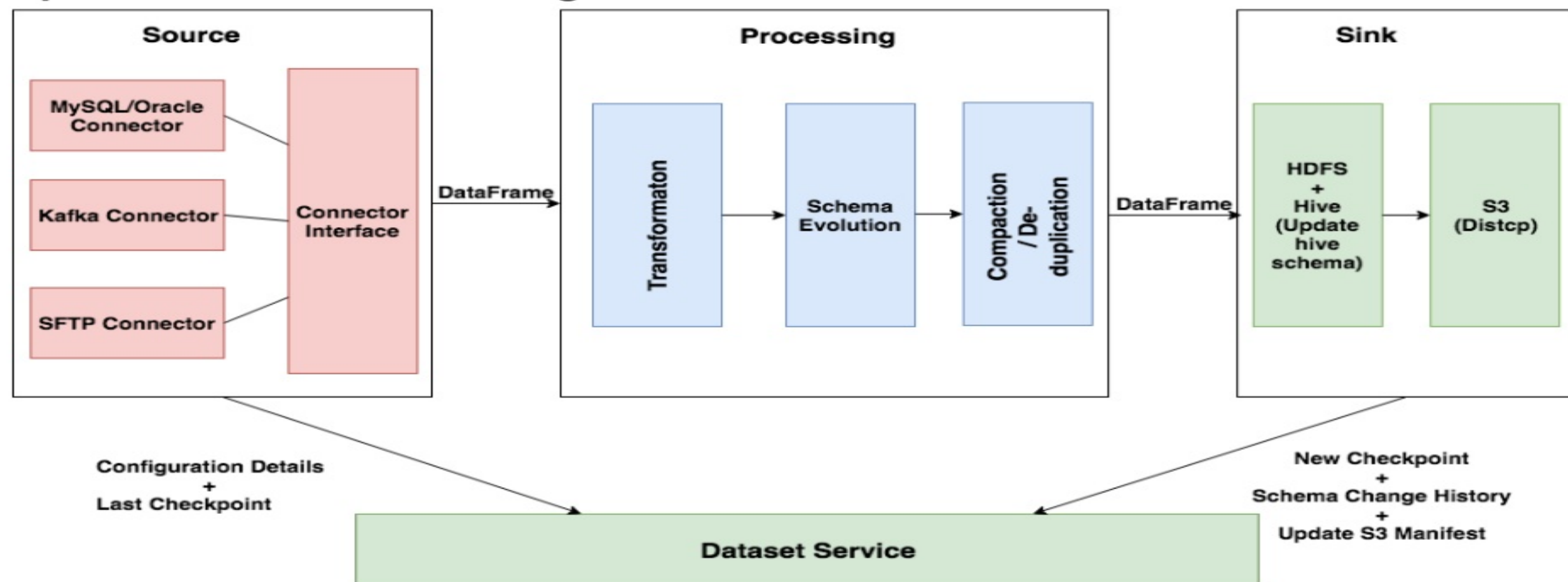
## Spark Connector Based Ingestion Architecture



# Connector Implementation

- Connect to data source using connection properties
- Create dataframe from source (using last checkpoint state)
- Throttle based on batch size
- Capture new checkpoint
- Optionally capture max record time

## Spark Connector Based Ingestion Architecture





# Transformations

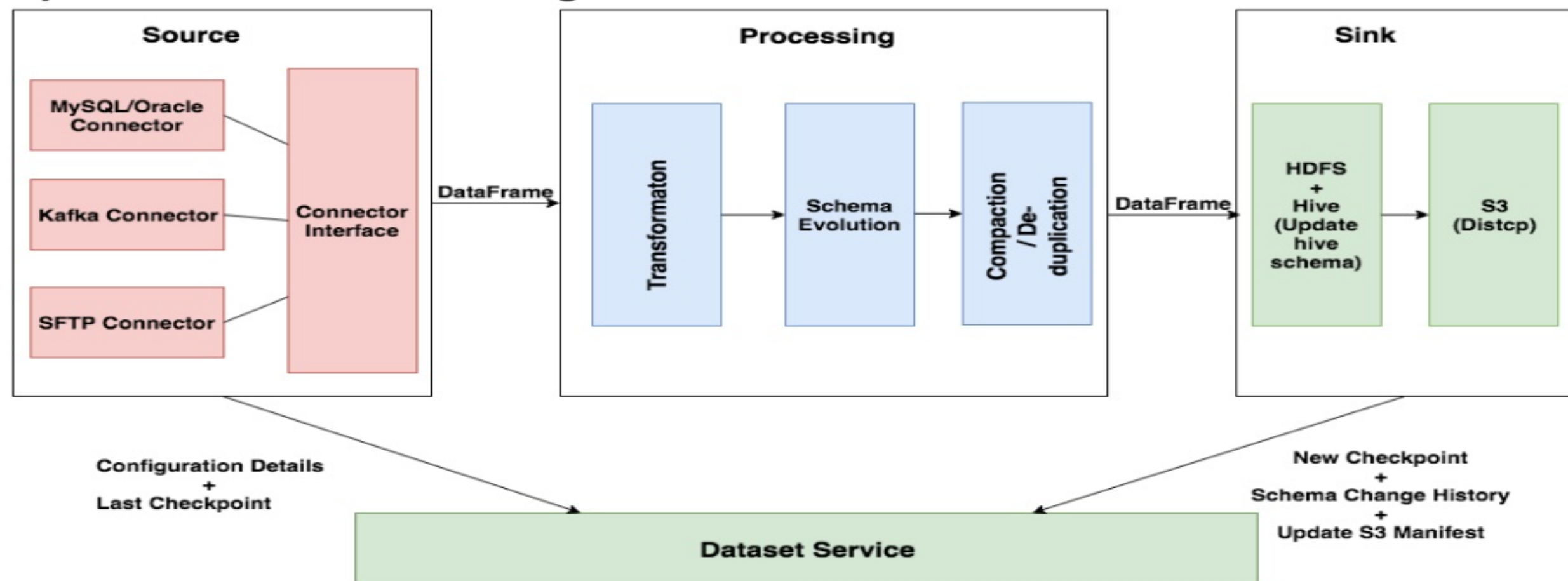
```
"df.withColumn("update_at",to_timestamp(col("update_at")))  
  .withColumn("id", col("id").cast(LongType))  
  .withColumn("ticket_id",col("ticket_id").cast(LongType))"
```

```
"df.withColumn("phone_no", md5(col("phone_no")))"
```

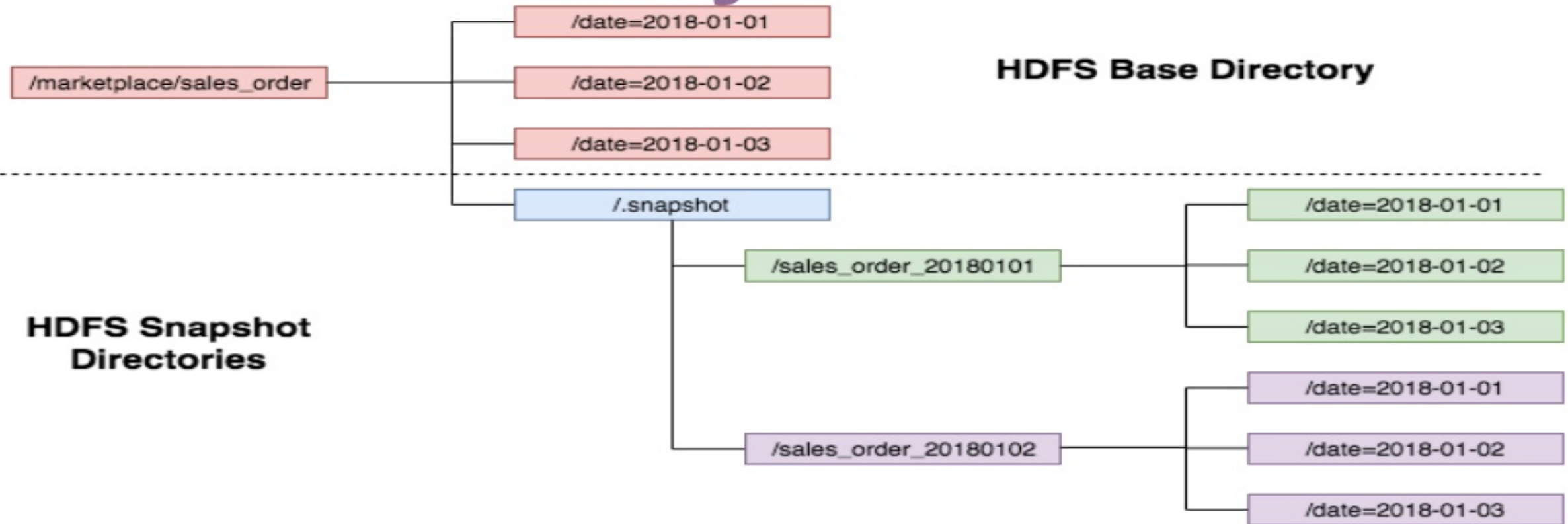
- Leverage Scala's Tool Box API
  - Converts to AST
  - Compiles and Runs AST



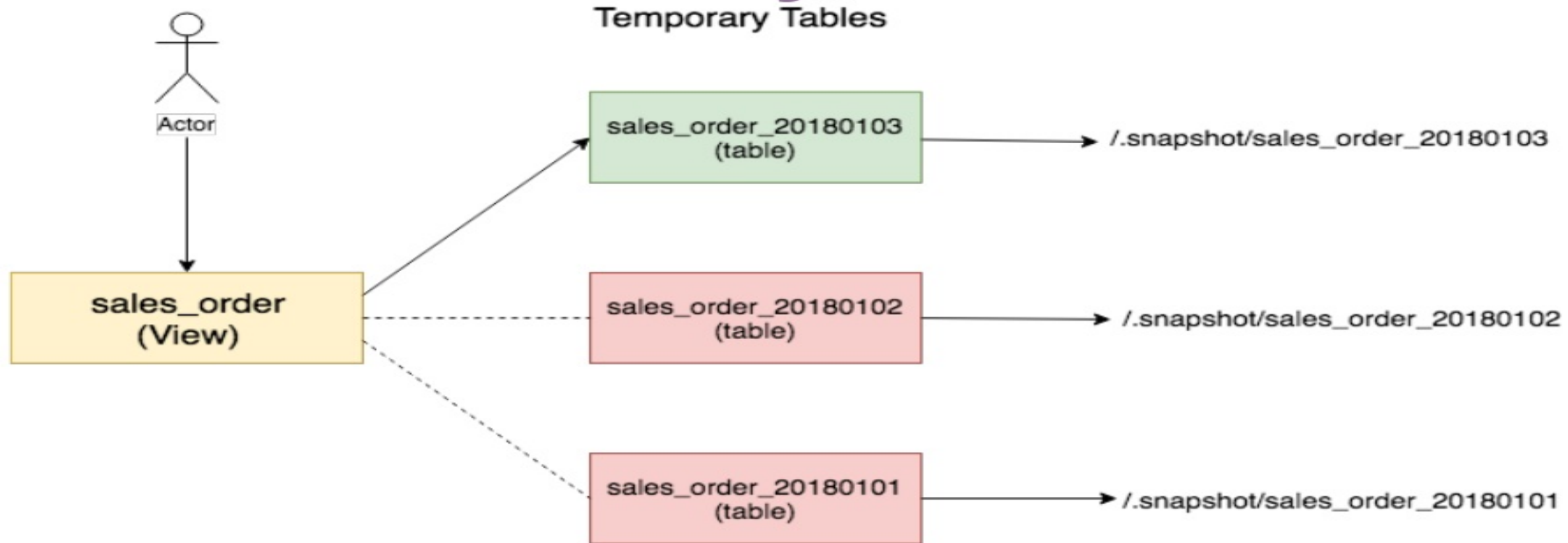
## Spark Connector Based Ingestion Architecture



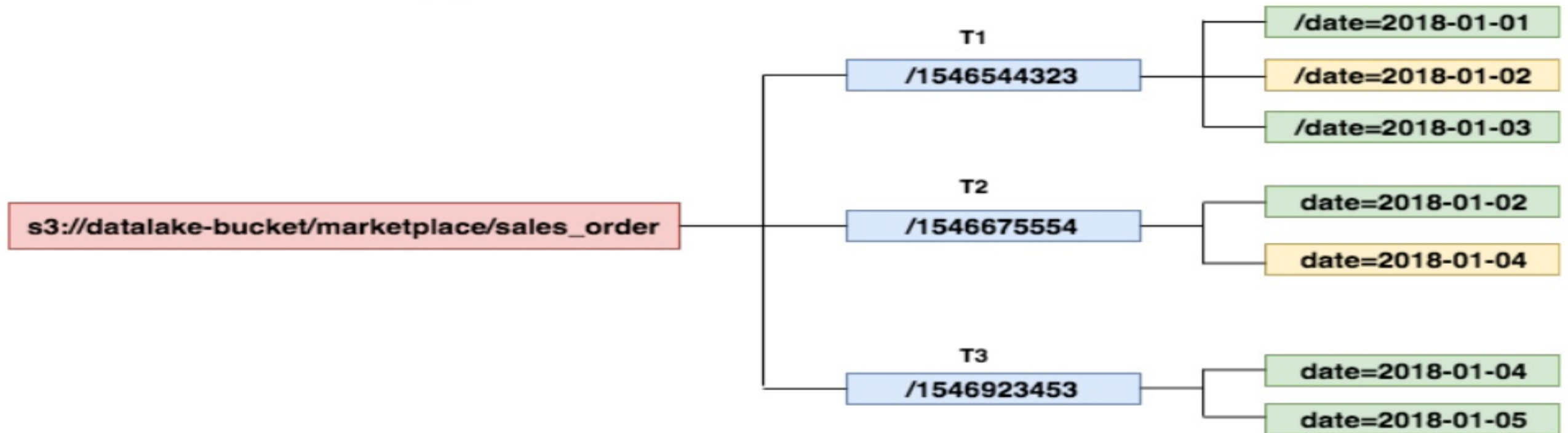
# HDFS Directory Structure




# Hive Schema Sync

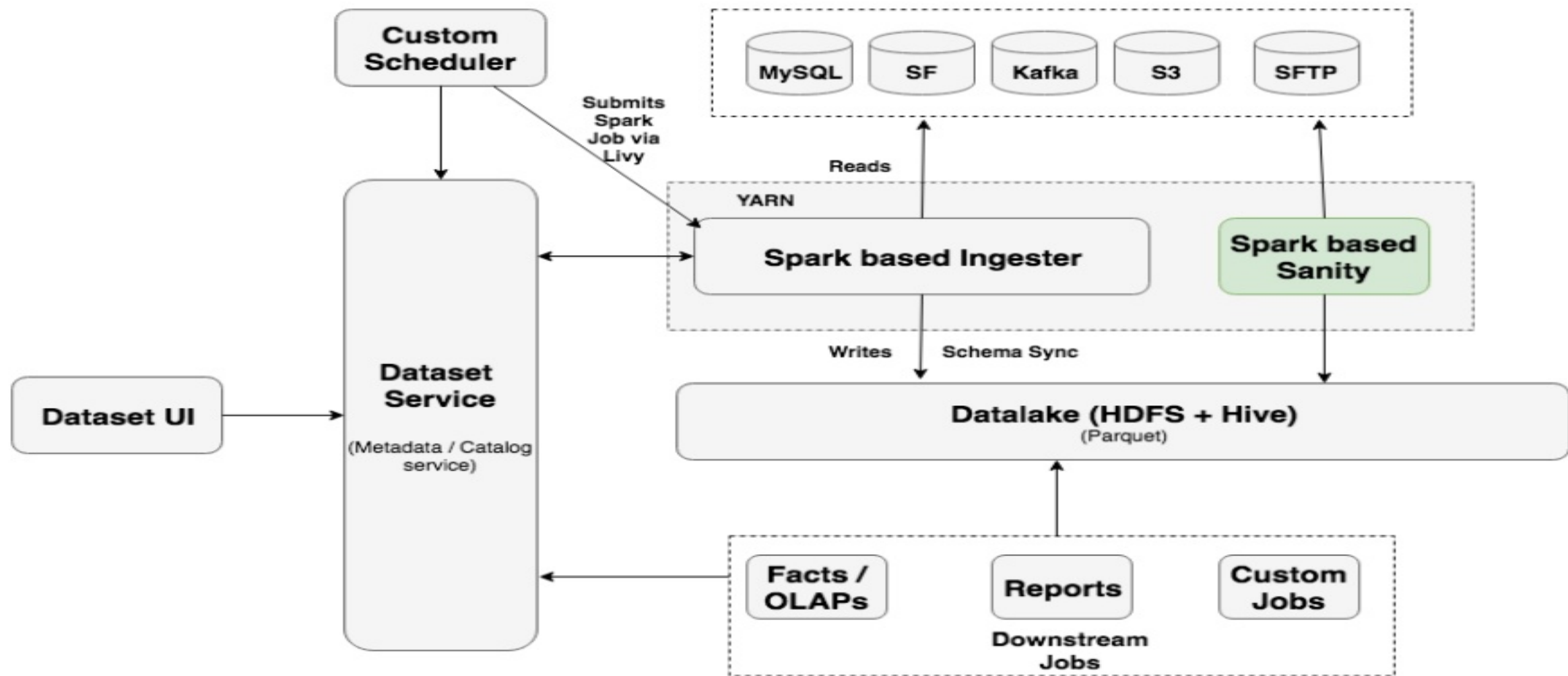


# S3 Storage Structure



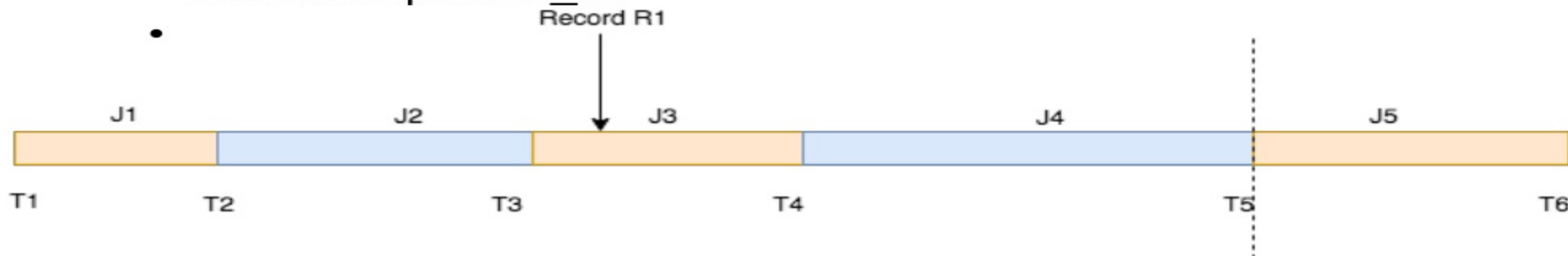
# Dataset Service S3 Manifest API

```
{   
  "date=2018-01-01": "s3://datalake-bucket/marketplace/sales_order/1546544323/date=2018-01-01",  
  "date=2018-01-02": "s3://datalake-bucket/marketplace/sales_order/1546675554/date=2018-01-02",  
  "date=2018-01-03": "s3://datalake-bucket/marketplace/sales_order/1546544323/date=2018-01-03",  
  "date=2018-01-04": "s3://datalake-bucket/marketplace/sales_order/1546923453/date=2018-01-04",  
  "date=2018-01-05": "s3://datalake-bucket/marketplace/sales_order/1546923453/date=2018-01-05"  
}
```



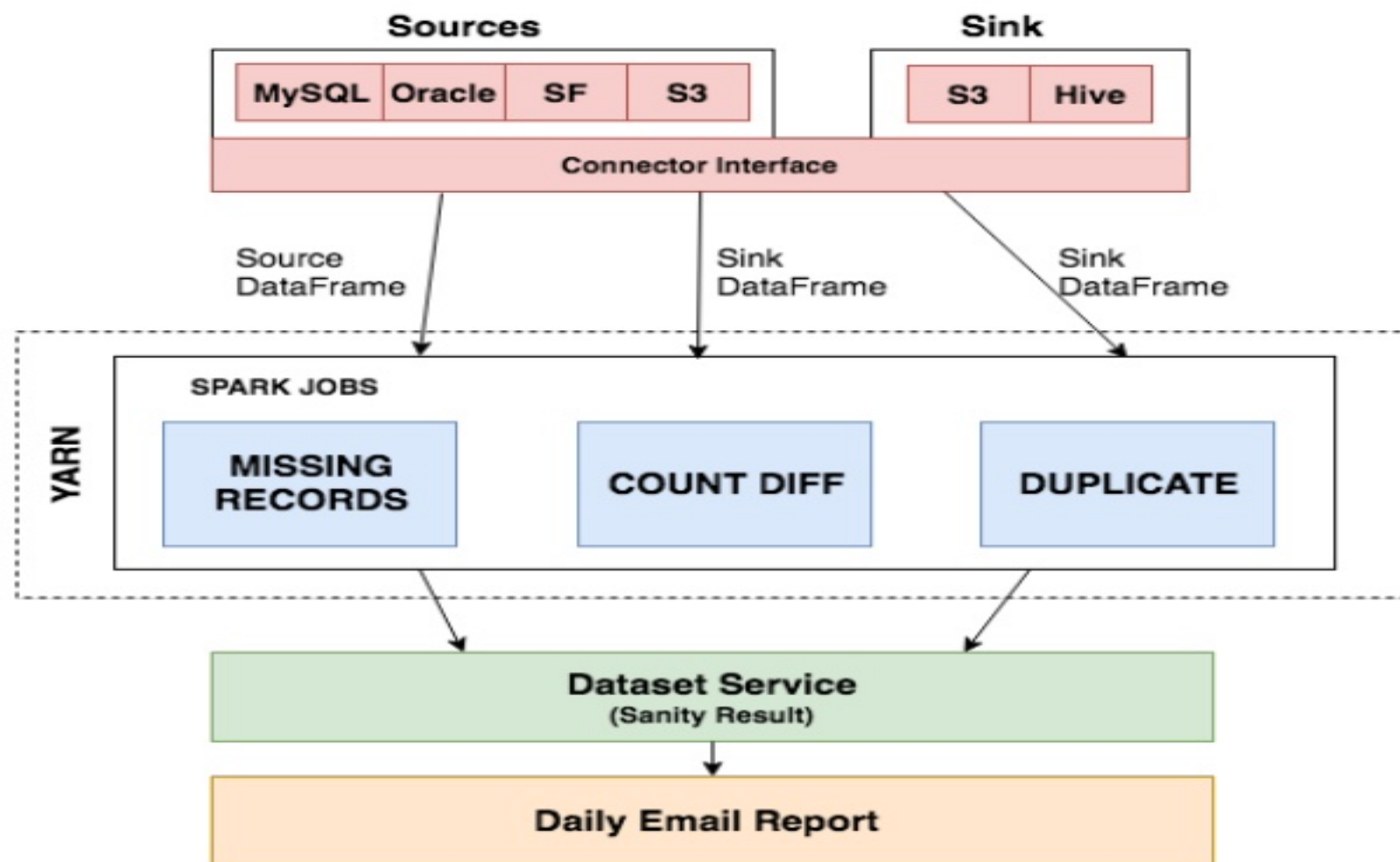
# Sanity

- Incomplete data ingestion
- Records with back date timestamp (updated\_at) are inserted in source table
- Incremental ingestion only ingest records with updated\_at > last max updated\_at
- 





# Sanity





# Reliability Factors

- One code base leveraging Spark DF abstraction
  - Simplifies monitoring, scheduling, transformations, schema evolution, compaction etc.
- Less code = Less Bugs
- Dependency Management via Record Time / Processing Time
- HDFS snapshots for read time consistencies
- S3 Manifest API for read time consistencies
- Sanity

# Thanks



[agrawalgagan@gmail.com](mailto:agrawalgagan@gmail.com)



@gaganagrawal24



/gaganagrawal24