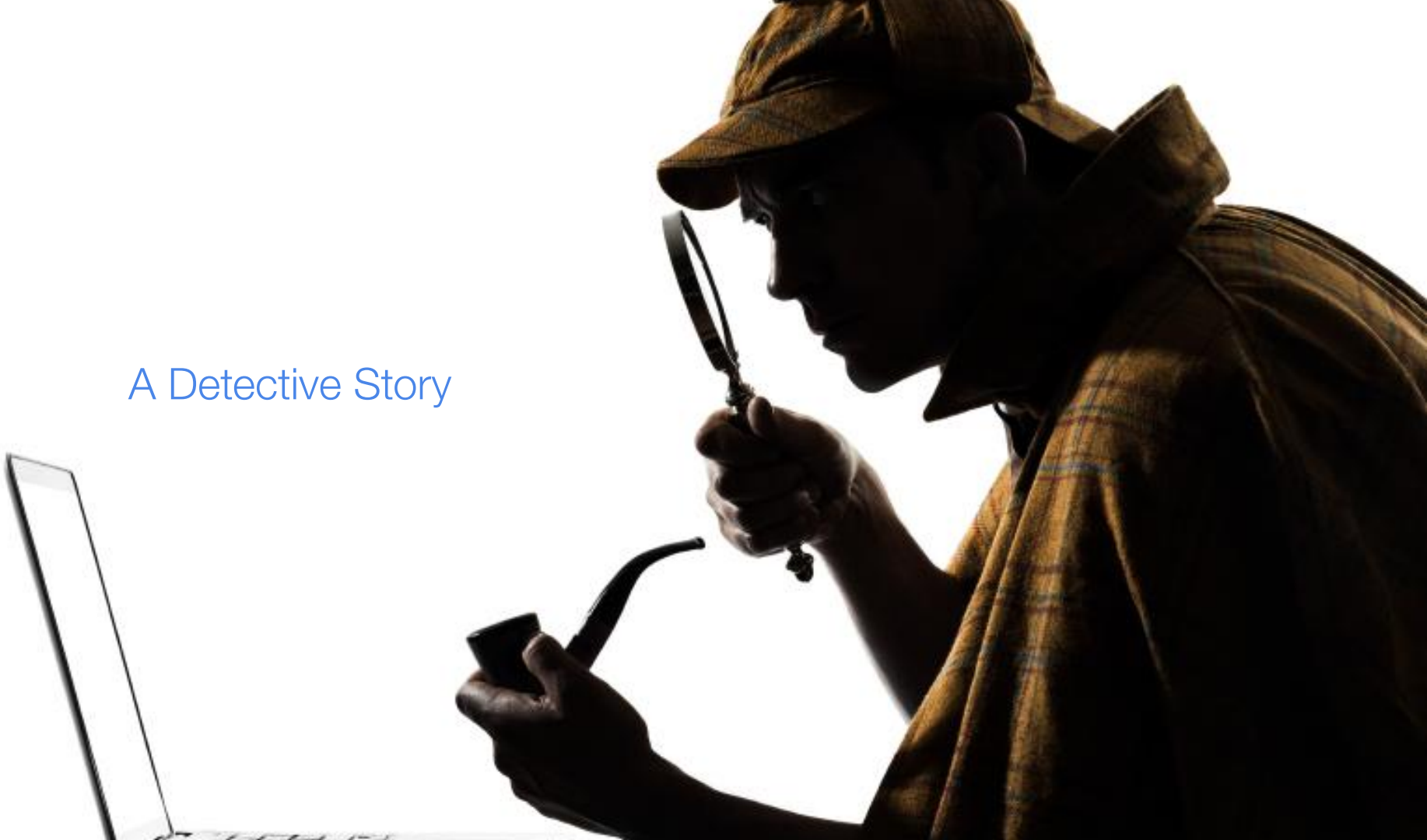
Cross-Platform Data Lineage with OpenLineage

Tracing lineage in Spark and Airflow

Who am I?

- Staff Software Engineer at Datakin
- Previously data infrastructure at Amazon and Cruise
- Clickstream data ingestion and analysis
- A/B test data analysis
- Data catalog

A Detective Story



A story - Mystery at Us Cooks

A story - Mystery at Us Cooks

A story - Mystery at Us Cooks

A story - Mystery at Us Cooks

Us Cooks Recommendations

Orders near you

Your neighborhood's latest orders

Kaffa Ethiopian Restaurant

2 items • $23.98 • 1 min ago 428

Veggie Grill - Downtown Seattle

3 items • $30.00 • 1 min ago 450

Sharetea - Renton

1 item • $8.25 • 1 min ago 7950

Today's offers

Aloha Plates

$1.99 Delivery Fee • 50–60 min 1244

Huckleberry Square Restaurant

$0.49 Delivery Fee • 35–45 min 5723

Taqueria los Potrillos - Seattle, ...

$2.99 Delivery Fee • 40–50 mi 1178

Us Cooks Recommendations

Metrics used to determine recommendations include

- Star ratings
- # of reviews
- Delivery time
- Repeat customers
- Regular and sale price

Menu Item Normalization

- Same menu items, different names
  - Pâté of roasted indigenous legumes, paired with a compote of seasonal berries, served on hearty sprouted wheat bread
  - Pan-roasted pastry rolls, layered with an herbed tomato puree, a creamy blend of artisanal cheeses, and tender bites of aged salami
  - Minced beef shoulder and caramelized onions in a sweet tomato and vinegar reduction served on a fresh brioche roll
- Restaurant rebranding
- Same chef, different theme

Goodhart's Law

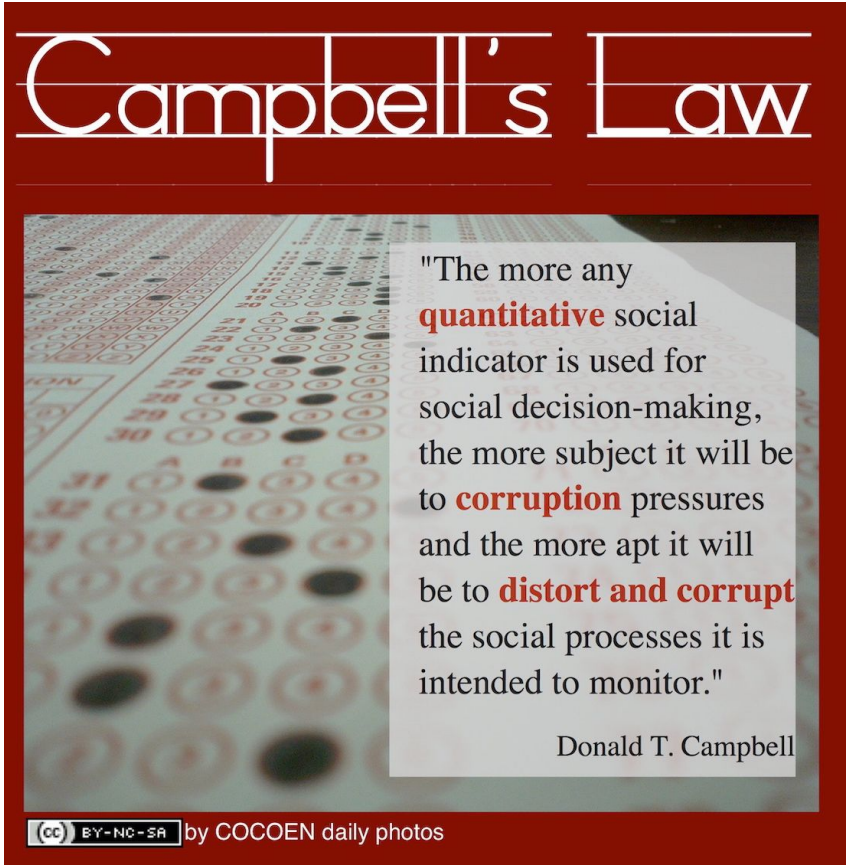
"Any observed statistical regularity
will tend to collapse
once pressure is placed upon it
for control purposes."

- Charles Goodhart

That is:

When a measure becomes a target,
it ceases to be a good measure.

Campbell's Law



Data Manipulation in Us Cooks

Algorithmically driving traffic to:

- New restaurants
- Highly reviewed restaurants
- Up and coming restaurants
- Popular orders
- Recommended orders

Leads to:

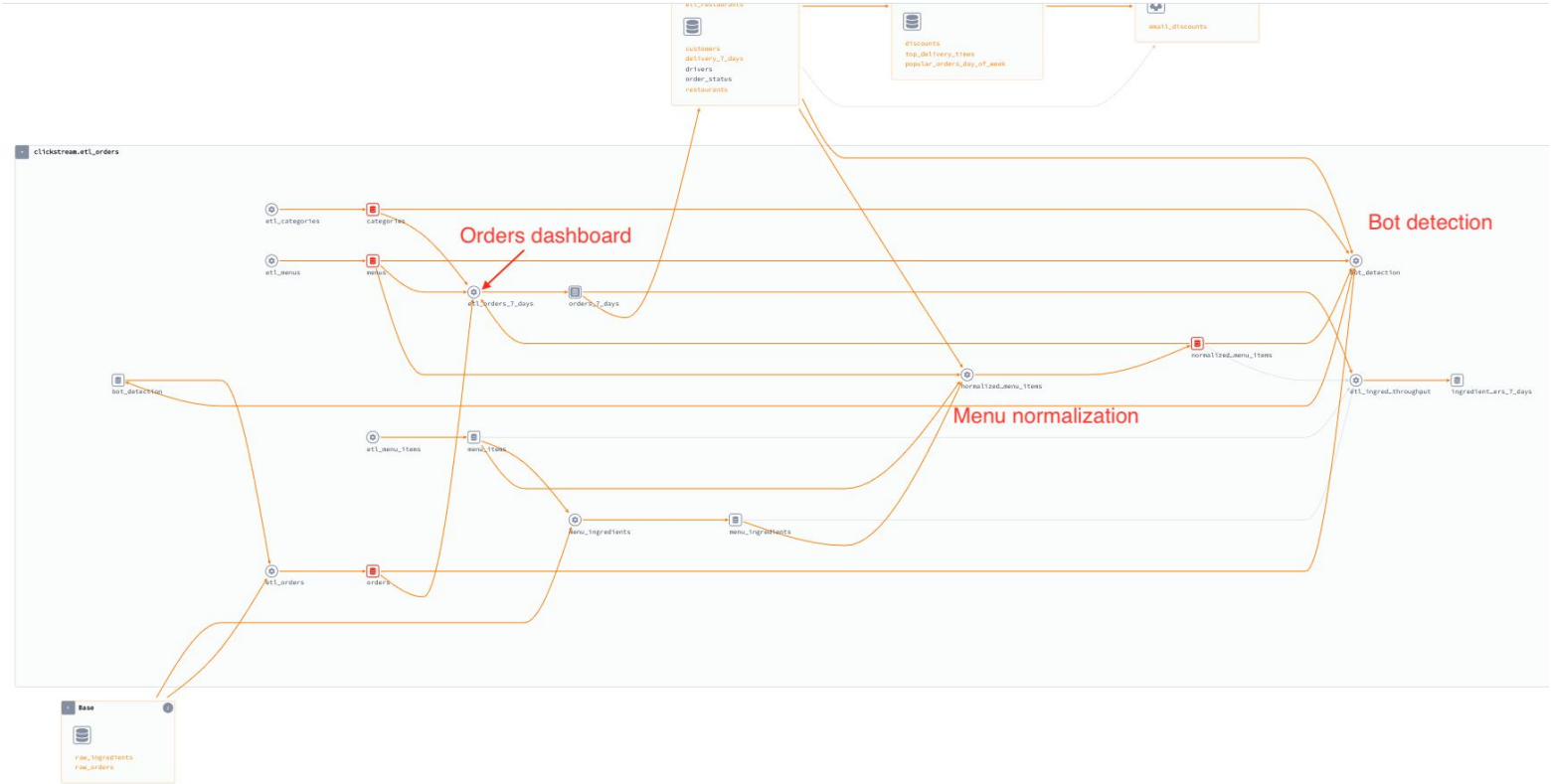
- Restaurants generating fake orders
- Restaurants leaving fake reviews
- Fake tweets and retweets
- Wholesalers generating fake reviews
- Fake review company services

Data Manipulation in Us Cooks

Counterattack:

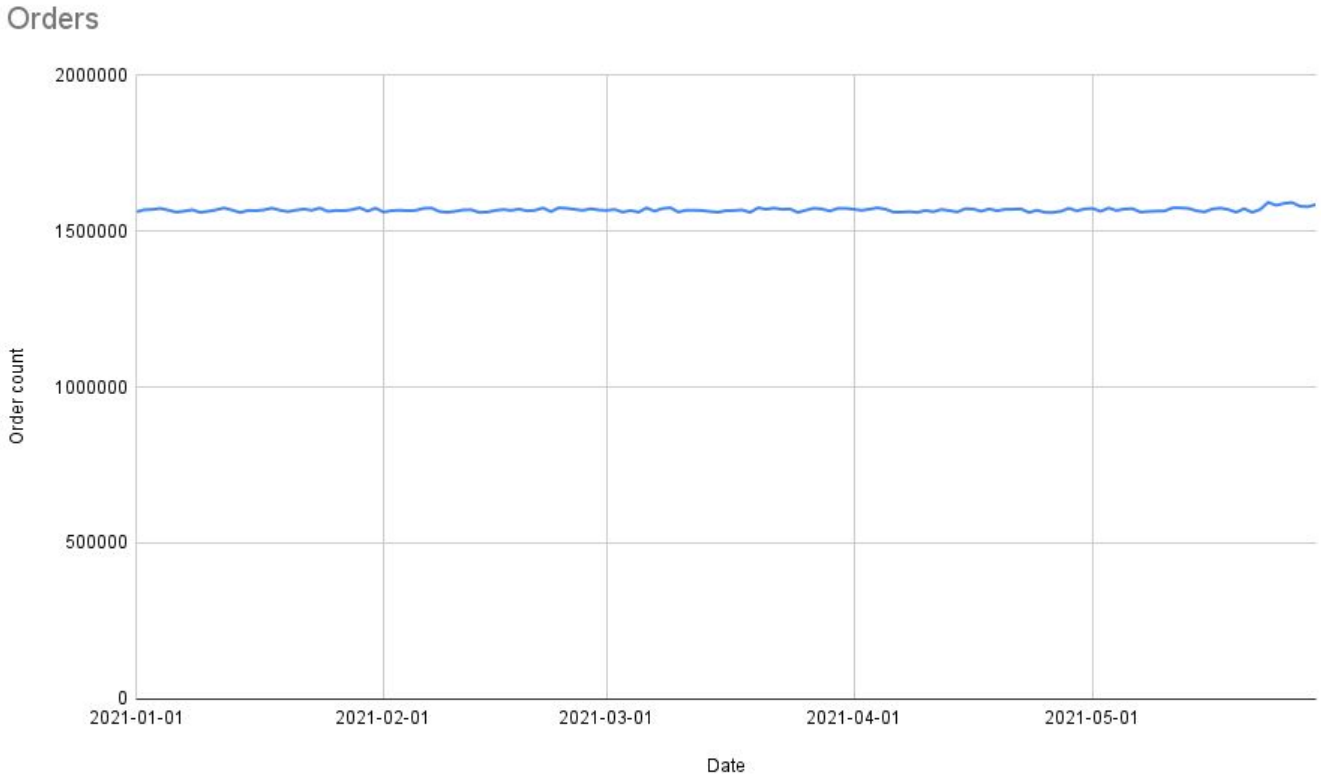
- De-fancification
- Bot detection
- Fake order detection
- Paid review detection
- Fake repeat customer detection

Data Dependency Graph

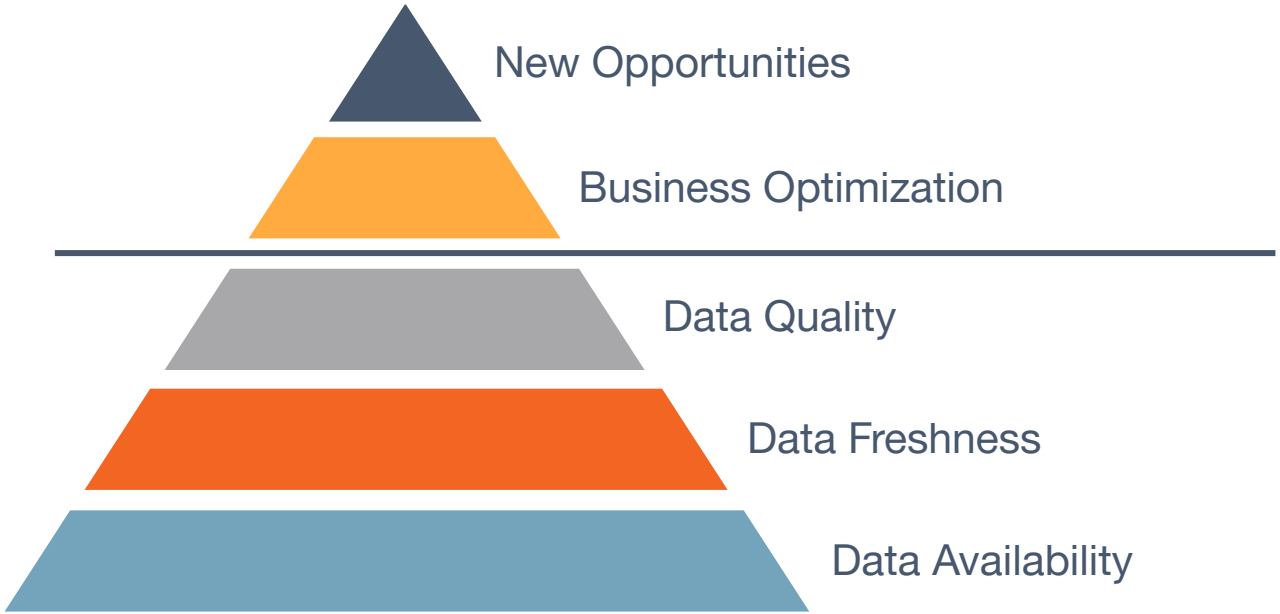


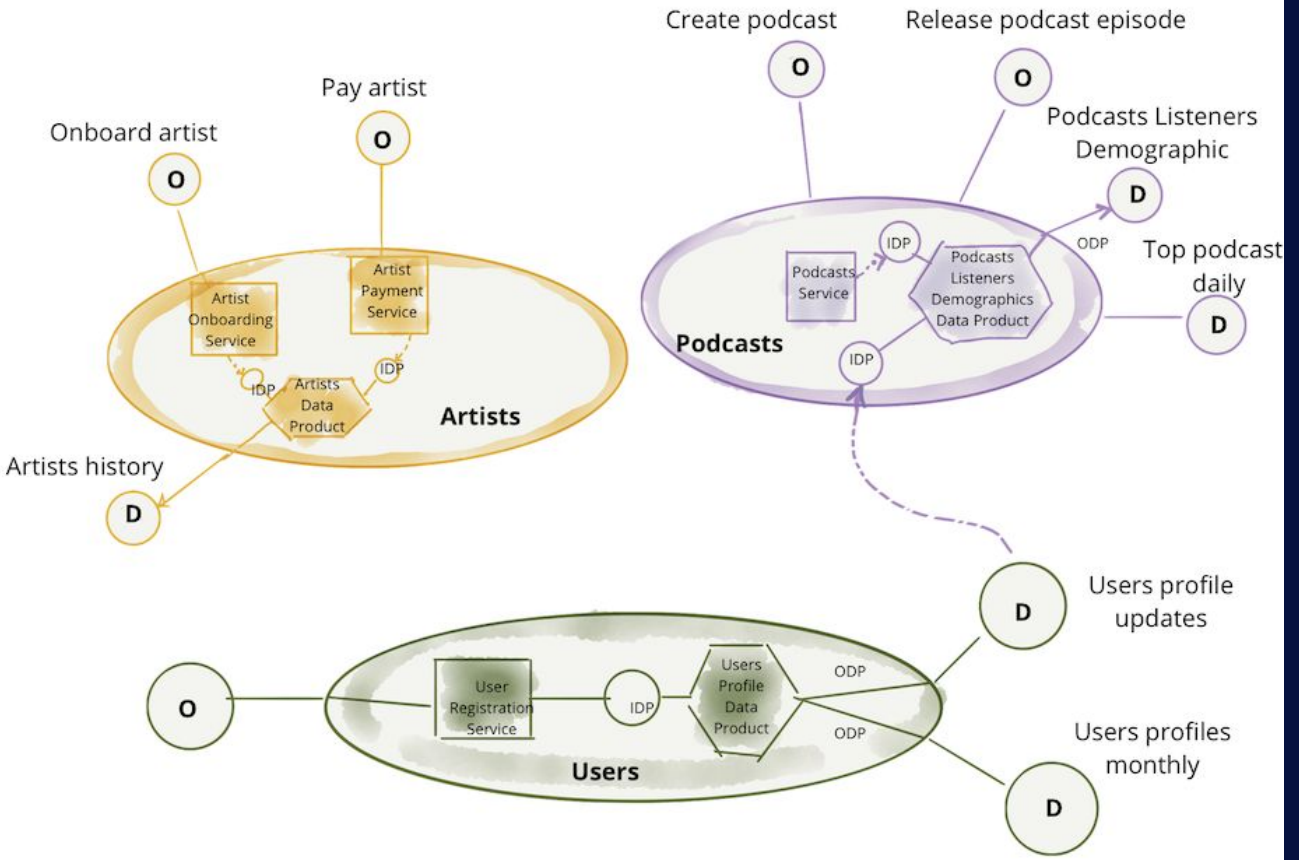
The culprit

The Fixed Chart



~~Maslow's~~ Data hierarchy of needs

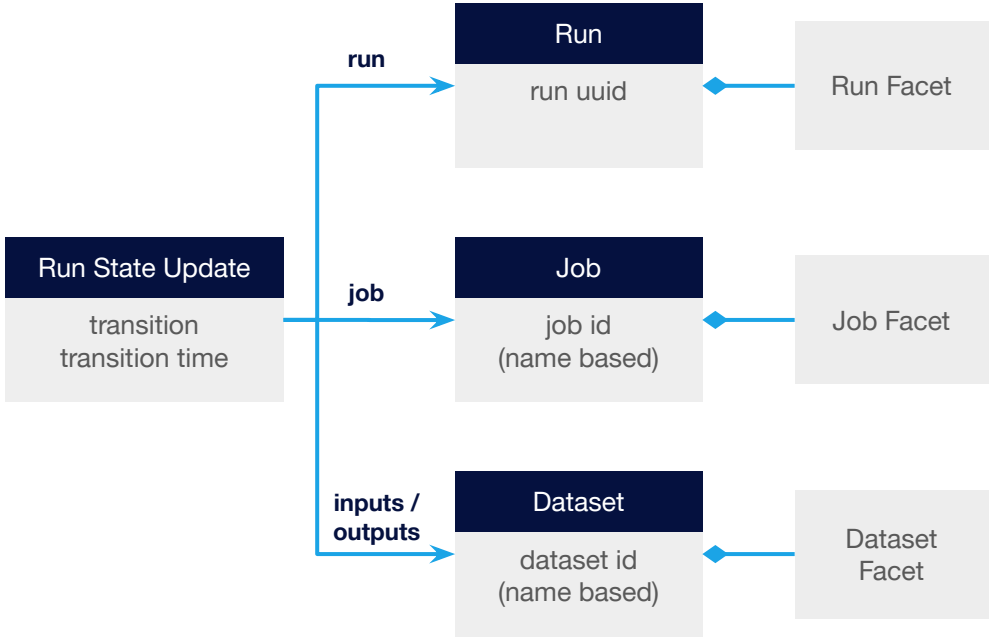




Lineage problems scale with the number of teams and interdependent datasets, NOT with the size of the data

OpenLineage Primer

Data Model



Built around core entities: Datasets, Jobs, and Runs

Defined as a JSONSchema spec

Consistent naming for:
Jobs (*scheduler.job.task*)
Datasets (*instance.schema.table)*

Jobs Hierarchy

- Jobs are hierarchical
  - Parent jobs may have no inputs or outputs
  - Child jobs inherit the namespace of their parents
- Examples:
  - Scheduler -> Task
  - Single application -> multiple operations
  - DAG -> Task -> External job

Facet Examples

Dataset:

- Input/Output statistics
- Schema
- Version

Job:

- Parent job
- Source code location
- Query plan
- Source control

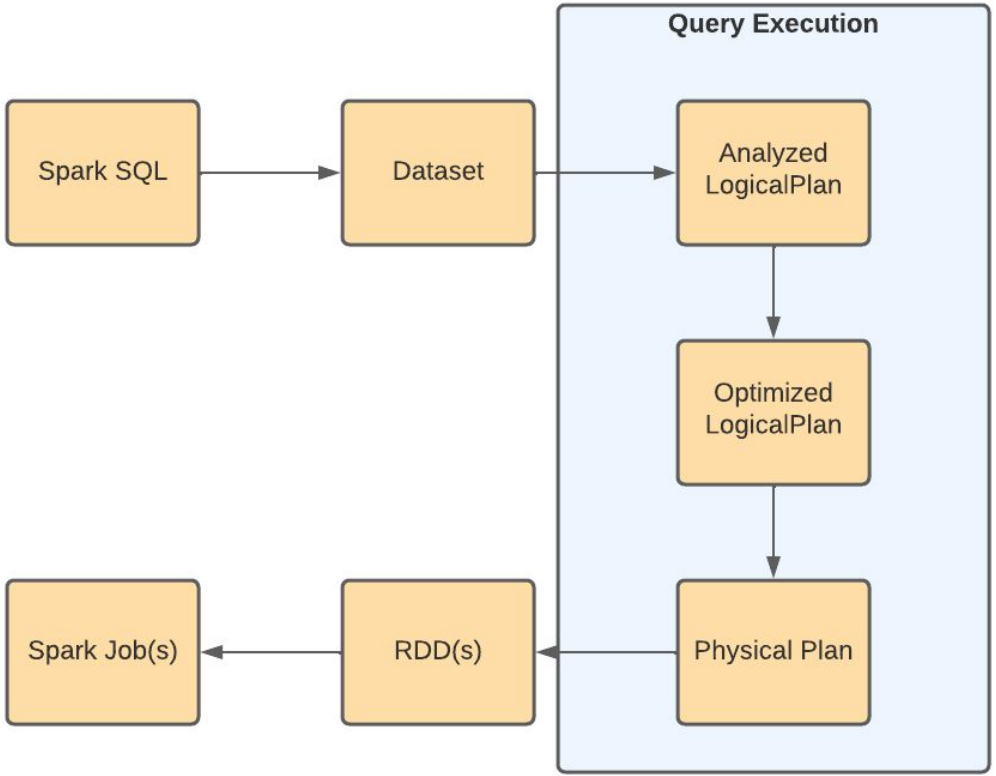
Run:

- Scheduled time
- Batch ID
- Parameters
- Cluster properties

Collecting Lineage

OpenLineage in Spark and Airflow

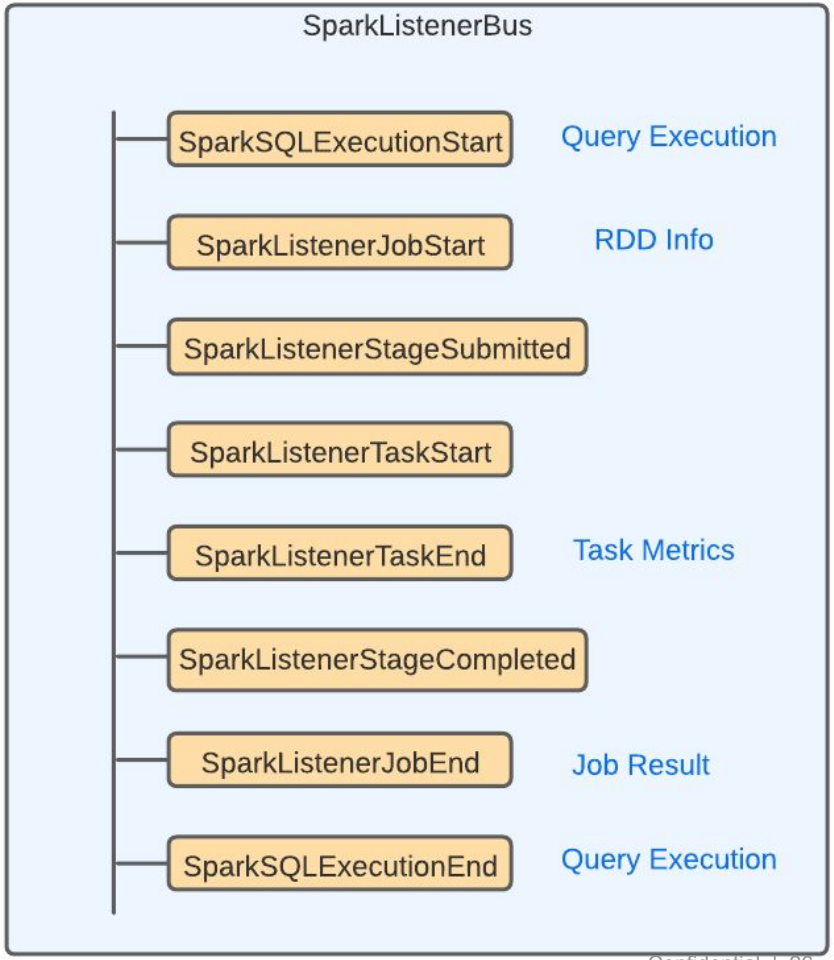
Spark SQL Query Execution



Spark ListenerBus

We implement the SparkListener interface and capture information from the Spark plan

- Serialized Optimized LogicalPlan
- Task input/output metrics
  - Bytes
  - Row counts
- Dataset Metadata
  - Location
  - Schema
  - Version (when available)
- Environment metadata
  - Databricks cluster properties



Configuring Spark

Configure the OpenLineageSparkListener

```
SparkSession.builder \
  .config('spark.jars.packages', 'io.openlineage:openlineage-spark:0.6.+')
  .config('spark.extraListeners', 'io.openlineage.spark.agent.OpenLineageSparkListener')
  .config('spark.openlineage.host', 'https://api.demo.datakin.com')
  .config('spark.openlineage.apiKey', 'your datakin api key')
  .config('spark.openlineage.namespace', '<NAMESPACE_NAME>')
   .getOrCreate()
```

Example Spark Job

```
spark = SparkSession\
  .builder \
  .appName("recommended_menu_items") \
  .getOrCreate()

orders = spark.read.parquet("gs://bq-airflow-spark/data/orders")
deliveries = spark.read.parquet("gs://bq-airflow-spark/data/top_delivery_times")

nexthour = datetime.now() + timedelta(hours=1)
(orders.join(deliveries, orders.id == deliveries.order_id, how='inner')
  .withColumn('hour', date_format(date_trunc('hour', from_unixtime('order_placed_on')), 'HH'))
  .filter(f"hour == '{nexthour.hour}'")
  .select(deliveries.menu_item_id, 'quantity')
  .groupBy('menu_item_id')
  .agg(sum('quantity').alias('quantity'))
  .sort(col('quantity').desc())
  .limit(20)
  .write.mode(saveMode='overwrite')
  .parquet('gs://bq-airflow-spark/data/upcoming_order_items'))

(spark.read.parquet('gs://bq-airflow-spark/data/upcoming_order_items')
 .write.format('jdbc')
 .mode(saveMode='overwrite')
 .option("url", "jdbc:postgresql://localhost:5432/spark")
 .option('driver', 'org.postgresql.Driver')
 .option("dbtable", "public.upcoming_order_items")
 .option("user", "spark_user")
 .option("password", "sparkpassword")
 .save())
```

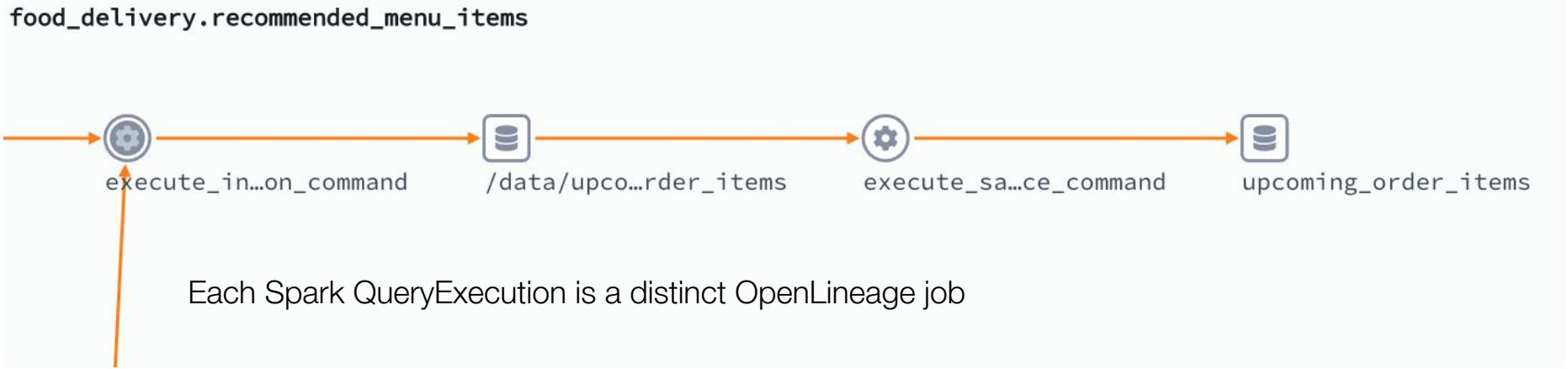
GCS sources

GCS out

and back in

Postgres output

Spark Jobs to OpenLineage



Spark Datasets supported

- Hadoop FS
  - HDFS
  - GCS
  - S3
  - ABFS
- JDBC (postgres, redshift, etc.)
- Hive
- BigQuery
- Snowflake
- Iceberg
- Kafka

Extensible API

- Custom LogicalPlan visitors
  - Extract dataset information from custom datasources
  - Extract custom dataset facets from plan nodes
- Custom dataset and facet builders
  - Receive SparkListener events and return custom datasets and facets
  - Generate custom job facets or run facets from application environment
- Java ServiceLoader mechanics
- Contribute to open source or keep them in house

Custom Datasets and Facets

- QueryPlanVisitor
- AbstractInputDatasetBuilder
- AbstractOutputDatasetBuilder
- CustomFacetBuilder

```
/**
 * {@link CustomFacetBuilder} that generates a {@link EnvironmentFacet} wh
 * Databricks.
 */
@Slf4j
public class DatabricksEnvironmentFacetBuilder
    extends CustomFacetBuilder<SparkListenerJobStart, EnvironmentFacet> {
  private HashMap<String, Object> dbProperties;
  private final OpenLineageContext openLineageContext;
  private Class dbutilsClass;
  private DbfsUtils dbutils;

  public DatabricksEnvironmentFacetBuilder(OpenLineageContext openLineageC
    this.openLineageContext = openLineageContext;
  }

  public static boolean isDatabricksRuntime() {
    return System.getenv().containsKey("DATABRICKS_RUNTIME_VERSION");
  }

  @Override
  protected void build(
      SparkListenerJobStart event, BiConsumer<String, ? super EnvironmentF
    consumer.accept(
        t: "environment-properties",
        new EnvironmentFacet(getDatabricksEnvironmentalAttributes(event)));
  }

  private HashMap<String, Object> getDatabricksEnvironmentalAttributes(
```

Airflow support

- Airflow 1.10.x

```
- from airflow import DAG

+ from openlineage.airflow import DAG
```

- Airflow 2.x

```
AIRFLOW__LINEAGE__BACKEND=openlineage.lineage_backend.OpenLineageBackend
```

- Airflow 2.3+

```
"airflow.plugins": ["OpenLineagePlugin = openlineage.airflow.plugin:OpenLineagePlugin"
```

Airflow Operator Support

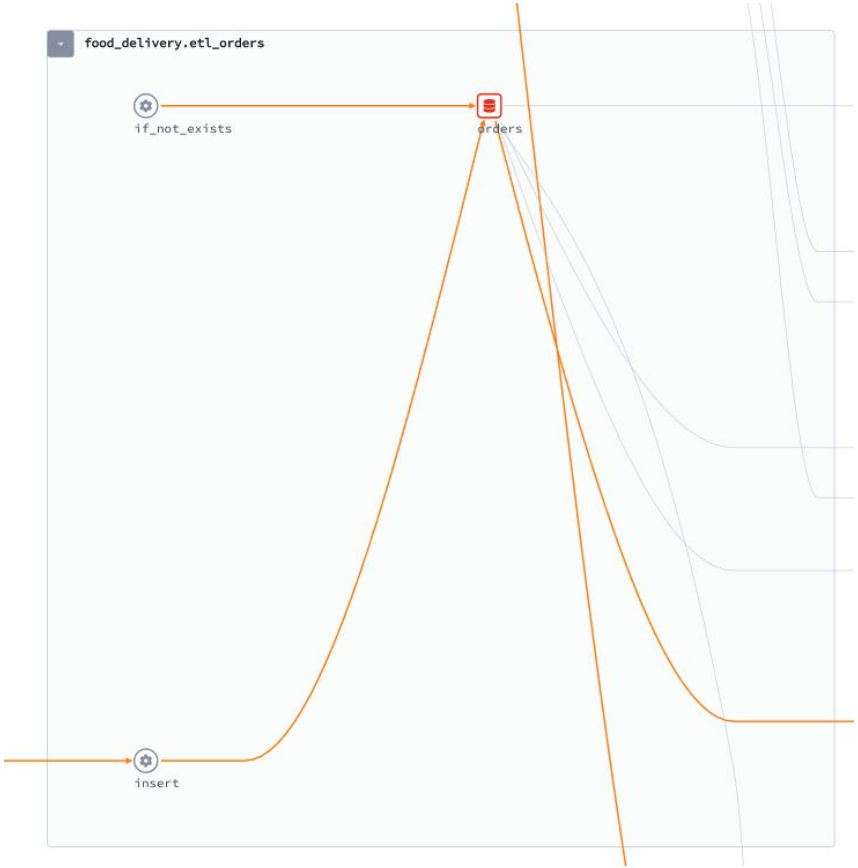
- Relies on lineage data exposed by the operators
  - BigQueryOperator
  - SnowflakeOperator
  - PostgresOperator
  - Custom Extractors
- PythonOperator and BashOperator unsupported

Example DAG

- DAG becomes a parent job (no inputs/outputs)
- Tasks are jobs with inputs/outputs
- Task job names are prefixed with DAG name

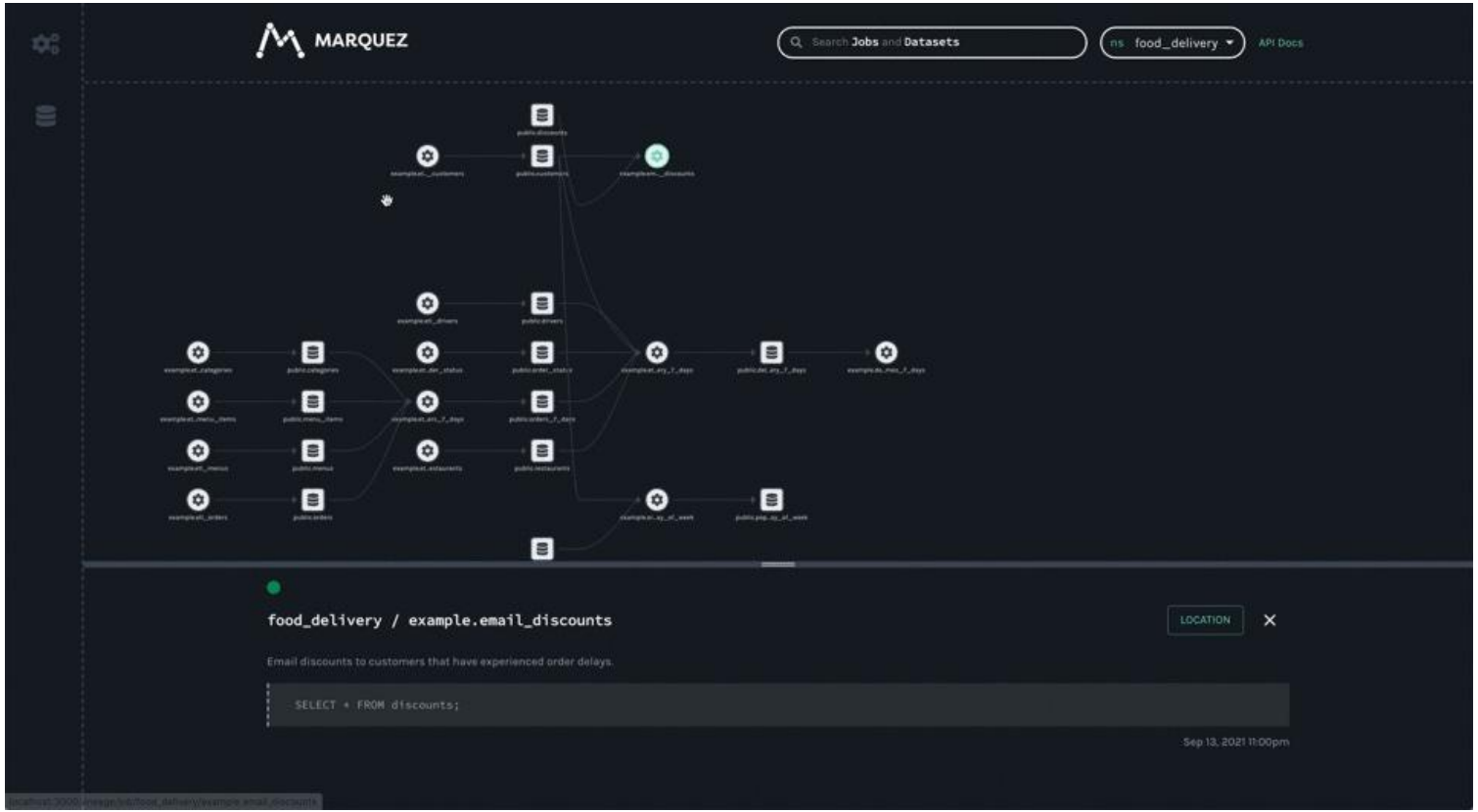
```
dag = DAG(
    'etl_orders',
    schedule_interval='@hourly',
    catchup=False,
    default_args=default_args,
    description='Loads newly placed orders daily.'
)

t1 = BigQueryOperator(
    task_id='if_not_exists',
    sql='''
    CREATE TABLE IF NOT EXISTS food_delivery.orders (
      id           INT64,
      placed_on    TIME,
      menu_item_id INT64,
      quantity     INT64,
      discount_id  INT64,
      comment      STRING
    )
    ''',
    use_legacy_sql=False,
    dag=dag
)

t2 = BigQueryOperator(
    task_id='insert',
    sql='''
    SELECT id, placed_on, menu_item_id, quantity, discount_id, comment
      FROM food_delivery.tmp_orders;
    '''
```

Example DAG

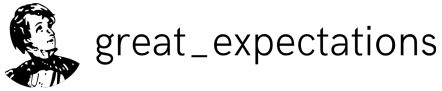


Marquez: open source metadata

Marquez: open source metadata



OpenLineage Contributors



Resources

- https://openlineage.io/
- https://github.com/OpenLineage/OpenLineage
- https://openlineage.slack.com/
- https://twitter.com/OpenLineage
- https://datakin.com/blog/
- https://marquezproject.github.io/marquez/

michael-collado-80351753

@PeladoCollado

Subsurface LIVE

The Cloud Data Lake Conference

Q&A