Spark, Delta Lake & Databricks

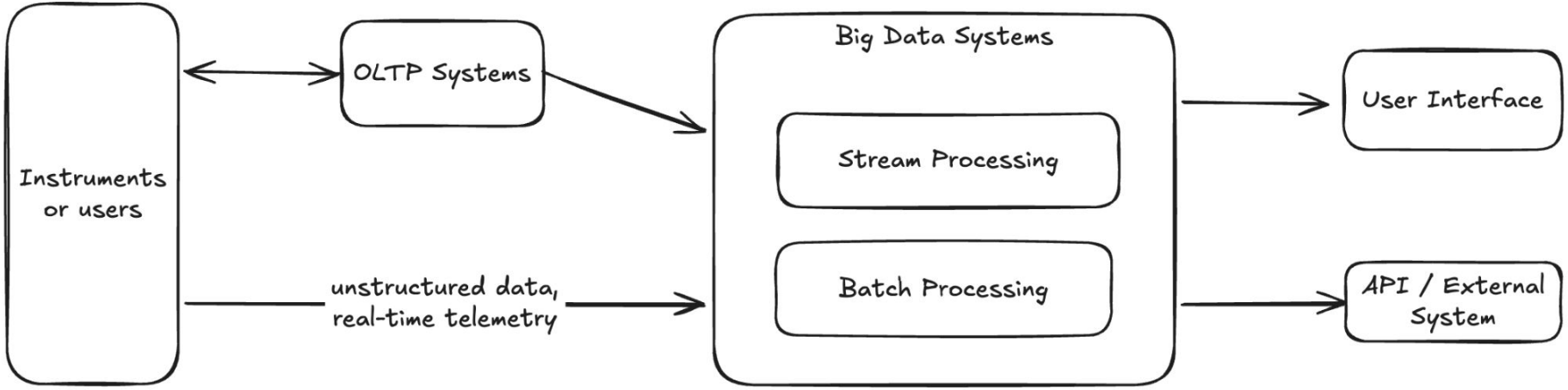
BI-BIG, 5.12.2024, Jan Lukány

DATAMOLE

DATAMOLE

datamole.ai

*a Czech **data and AI company** of **80+ people** founded in **2015.** We **develop custom AI, IoT & UI solutions** that innovate industrial companies worldwide – in agriculture, machinery or biotech.*


Instruments or users
OLTP Systems
unstructured data, real-time telemetry
Big Data Systems
Stream Processing
Batch Processing
User Interface
API / External System

DATAMOLE

Big Data Challenges

5Vs

- Velocity is the speed at which the data is created and how fast it moves.
- Volume is the amount of data qualifying as big data.
- Value is the value the data provides.
- Variety is the diversity that exists in the types of data.
- Veracity is the data's quality and accuracy.

This lecture focuses on:

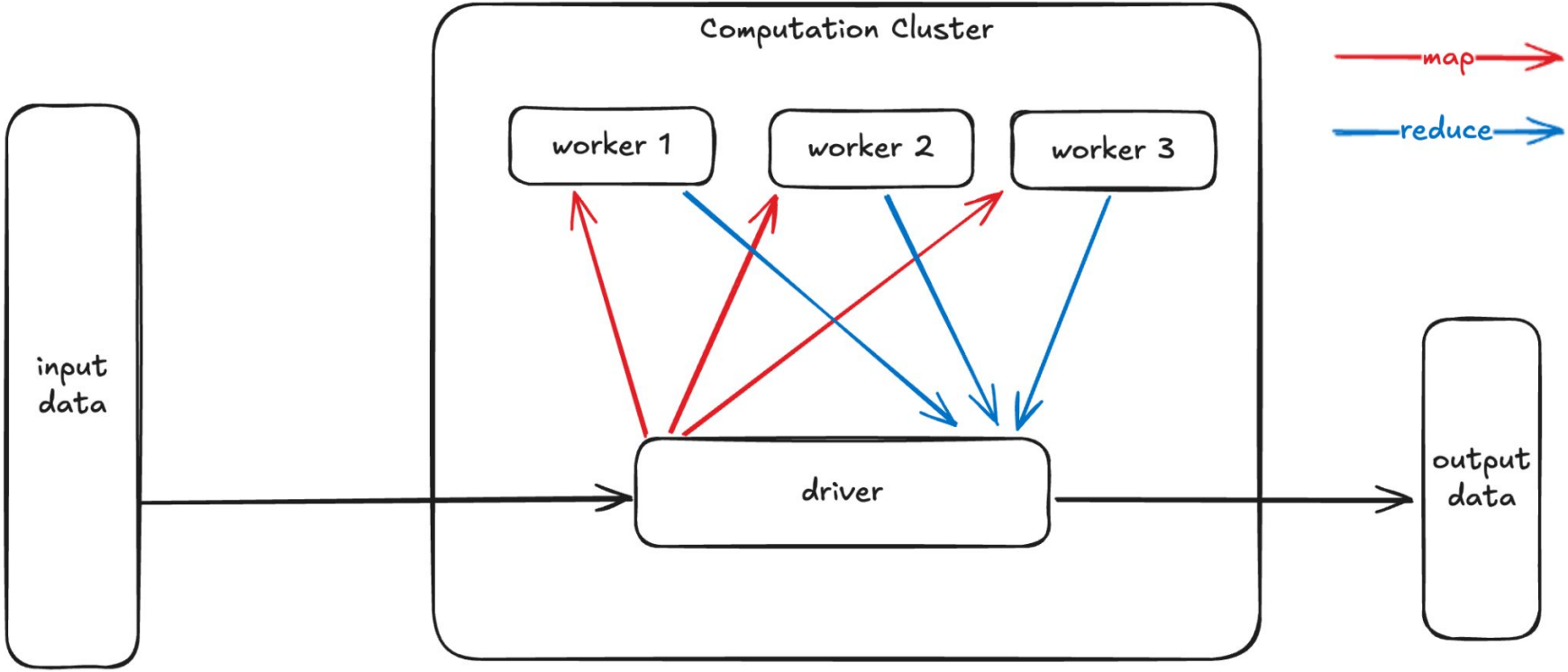
- How to efficiently process the data?
- How to reliably store the data?

Outline

- Apache Spark – Distributed Processing Framework
  - Open-source
- Delta Lake – Reliable Storage Layer
  - Open-source
- Databricks – Data Intelligence Platform
  - Built on top of Apache Spark and Delta Lake

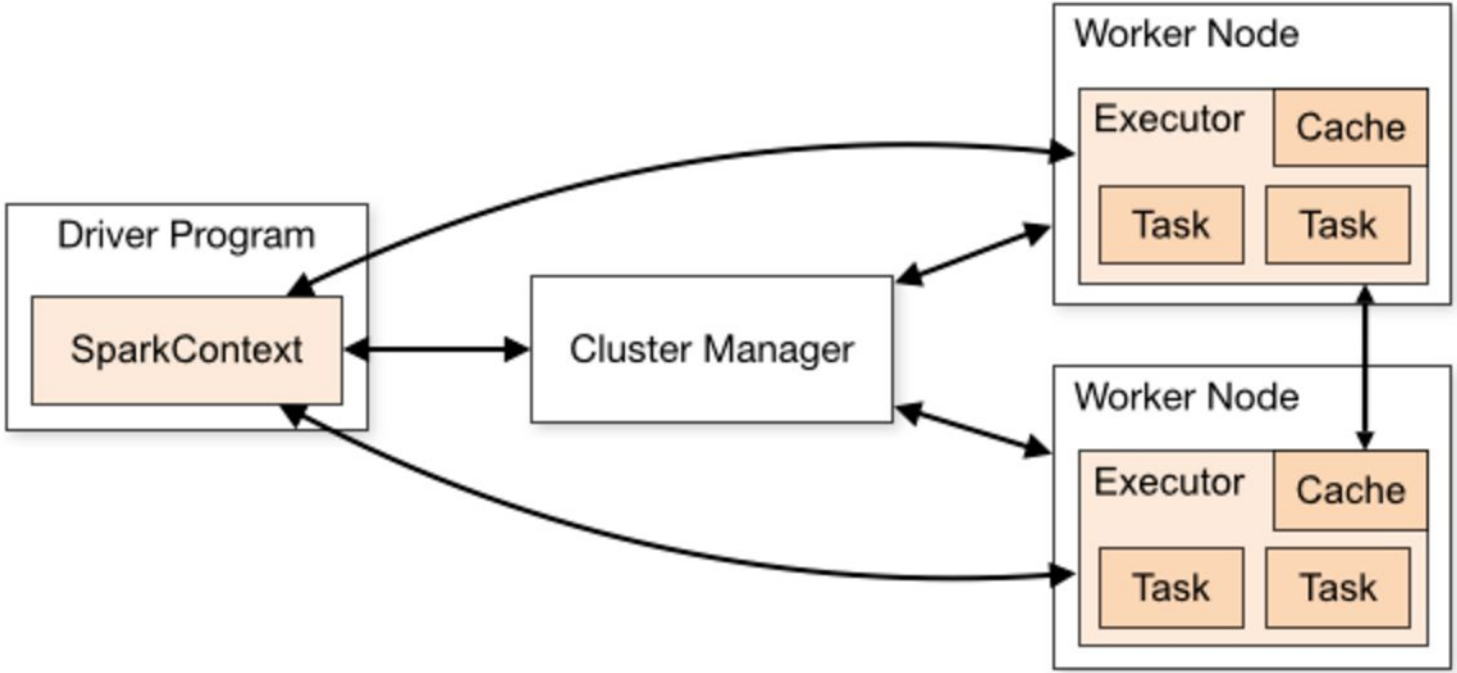
DATAMOLE

Apache Spark: map-reduce simplified



DATAMOLE

Apache Spark: architecture



Apache Spark: architecture

- RDD (Resilient Distributed Dataset)
  - Immutable collection of objects
  - Fault-tolerant: Automatically recomputed in case of node failure
  - Low-level API for transformations (e.g., map, filter)
- DataFrame
  - Abstraction on RDDs
  - Distributed collection of data as named columns
  - API in multiple languages: Python, Scala, Java, R
- DAG (Directed Acyclic Graph)
  - Execution plan broken down into stages and tasks
  - Fault tolerance – recomputing failed tasks

Apache Spark: Transformations and Actions

- Transformations are lazy – not evaluated immediately
  - filter, groupby, map…
- Actions trigger jobs
  - collect, count, write

Apache Spark: batch processing example

e.g. 10 TB

```python
from pyspark.sql import SparkSession

# Create SparkSession
spark = SparkSession.builder \
    .appName("BatchProcessingExample") \
    .getOrCreate()

# Read a CSV file into a DataFrame
df = spark.read.csv("data/sales.csv", header=True, inferSchema=True)

# Perform transformations
result = df.groupBy("region").sum("sales")

# Write the output to a new file
result.write.csv("output/region_sales.csv", header=True)
```

DATAMOLE

Apache Spark: structured streaming

```python
from pyspark.sql import SparkSession

# Create SparkSession
spark = SparkSession.builder \
    .appName("StreamProcessingExample") \
    .getOrCreate()

# Read streaming data from Kafka
stream_df = spark.readStream \
    .format("kafka") \
    .option("kafka.bootstrap.servers", "localhost:9092") \
    .option("subscribe", "sales") \
    .load()

# Parse the Kafka data
sales_data = stream_df.selectExpr("CAST(value AS STRING)")

# Perform transformations
result = sales_data.groupBy("region").sum("sales")

# Write the streaming output to console
query = result.writeStream \
    .outputMode("complete") \
    .format("console") \
    .start()

query.awaitTermination()
```

DATAMOLE

Apache Spark: example integrations

- File-based (CSV, JSON, Parquet, XML…)
- Databases and Warehouses
  - JDBC (MySQL, PostgreSQL, Oracle, SQL Server)
  - NoSQL (MongoDB, Cassandra, Redis)
  - Cloud Warehouses (Amazon Redshift, Google BigQuery)
- Message and Event Streaming
  - Apache Kafka, Amazon Kinesis, Azure EventHub, RabbitMQ
- **Distributed File Systems**
  - HDFS, Amazon S3, Azure Blob Storage, Google Cloud Storage
- Delta Lake and Table Formats
  - **Delta Lake**
  - Apache Iceberg
  - Hudi

DELTA LAKE

DATAMOLE

Delta Lake

Key Features

ACID Transactions
Protect your data with serializability, the strongest level of isolation

Scalable Metadata
Handle petabyte-scale tables with billions of partitions and files with ease

Time Travel
Access/revert to earlier versions of data for audits, rollbacks, or reproduce

Open Source
Community driven, open standards, open protocol, open discussions

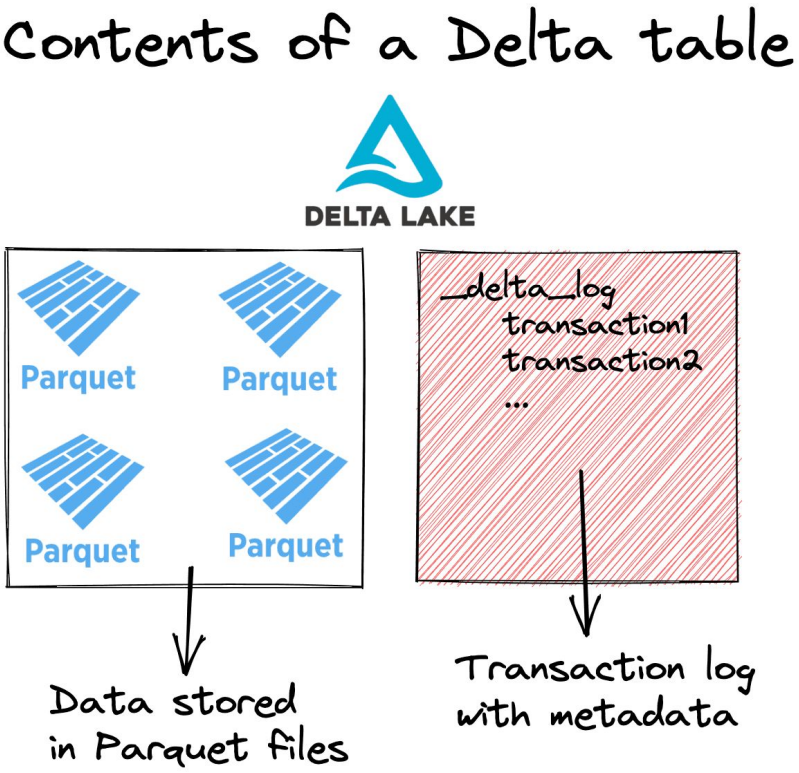
Unified Batch/Streaming
Exactly once semantics ingestion to backfill to interactive queries

Schema Evolution / Enforcement
Prevent bad data from causing data corruption

Audit History
Delta Lake log all change details providing a fill audit trail

DML Operations
SQL, Scala/Java and Python APIs to merge, update and delete datasets

Delta Lake – Delta Table



Delta Lake – Demo

01-delta-lake.ipynb

Delta Lake – on cheap cloud storages

- Azure Blob Storage
- Amazon S3
- Google Cloud Storage

Delta Lake – ACID

- **A**tomicity – process all or nothing
- **C**onsistency – always valid state
- **I**solation – read while writing, safe multiple writes
- **D**urability – transactions are permanent

Guarantees **over one table only**

Delta Lake – scalability

- Data file skipping using statistics (e.g. min/max)
- Columnar format of parquets
  - Read only parts of data files
- Data clustering
  - Storing similar data close together
  - Liquid clustering, Z-Ordering, partitioning

Delta Lake – time travel

- transaction log
- selecting specific version from the log
- optimize and vacuum operations
  - optimize – compact multiple small files into large ones
  - vacuum – delete unused data files

```
df = spark.read.format("delta").option("versionAsOf", 0).load("data/delta_census")
```

Delta Lake – Change Data Feed

- Tracking row-level changes for updates and deletes
  - no need for append-only (no update/delete) – transaction log is enough

```sql
ALTER TABLE myDeltaTable SET TBLPROPERTIES (delta.enableChangeDataFeed = true)
```

```python
# providing a starting version
spark.readStream.format("delta") \
  .option("readChangeFeed", "true") \
  .option("startingVersion", 0) \
  .table("myDeltaTable")
```

DATAMOLE

Delta Lake – schema evolution and enforcement

- allow adding new columns and changing (some) data types
- prevent bad data from being written

Delta Lake – Unified Batch and Streaming

- Batch – historical data processing in bulk
- Streaming – real-time data processing
- Same API for both


APACHE
Spark
TM

&

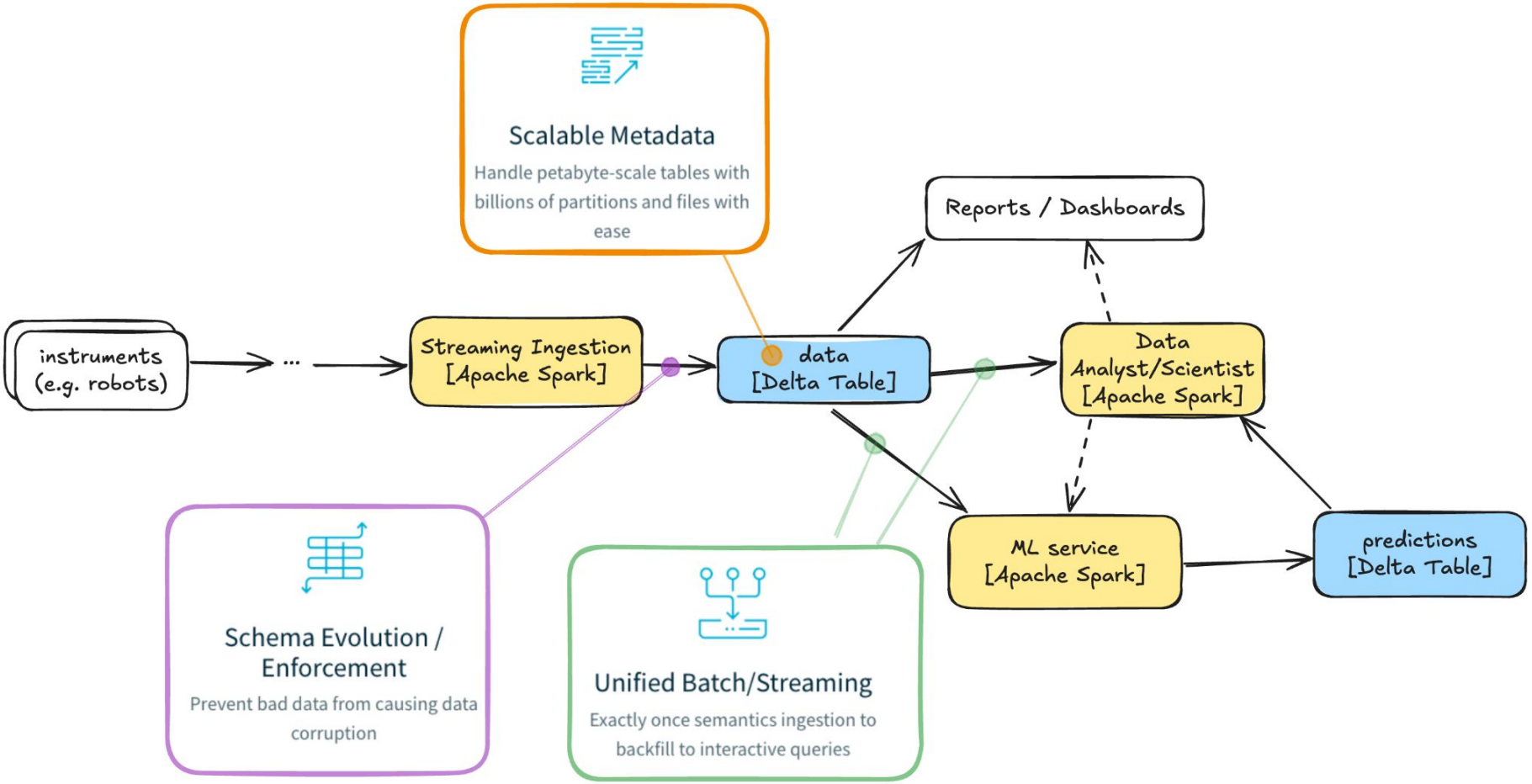

DELTA LAKE

Example Batch Processing

```python
# Read data from Delta table
input_df = spark.read.format("delta").load("input-table-path")

# Perform simple processing (e.g., filtering and adding a column)
processed_df = (
    input_df
    .filter(col("column_name") > 100)
    .withColumn("new_column", col("column_name") * 2)
)

# Write processed data back to a Delta table
(
    processed_df
    .write.format("delta")
    .mode("overwrite")
    .save("output-table-path")
)
```

Example Stream Processing

```python
# Read data from Delta table as a streaming source
input_stream = spark.readStream.format("delta").load("input-table-path")

# Perform simple processing (e.g., filtering and adding a column)
processed_stream = (
    input_stream.filter(col("column_name") > 100)
    .withColumn("new_column", col("column_name") * 2)
)

# Write processed data to Delta table as a streaming sink
query = (
    processed_stream.writeStream.format("delta")
    .outputMode("append")
    .option("checkpointLocation", "/path/to/checkpoint-dir")
    .start("output-table-path")
)
```


Scalable Metadata
Handle petabyte-scale tables with billions of partitions and files with ease
Reports / Dashboards
instruments (e.g. robots)
...
Streaming Ingestion [Apache Spark]
data [Delta Table]
Data Analyst/Scientist [Apache Spark]
ML service [Apache Spark]
predictions [Delta Table]
Schema Evolution / Enforcement
Prevent bad data from causing data corruption
Unified Batch/Streaming
Exactly once semantics ingestion to backfill to interactive queries

DATAMOLE

databricks

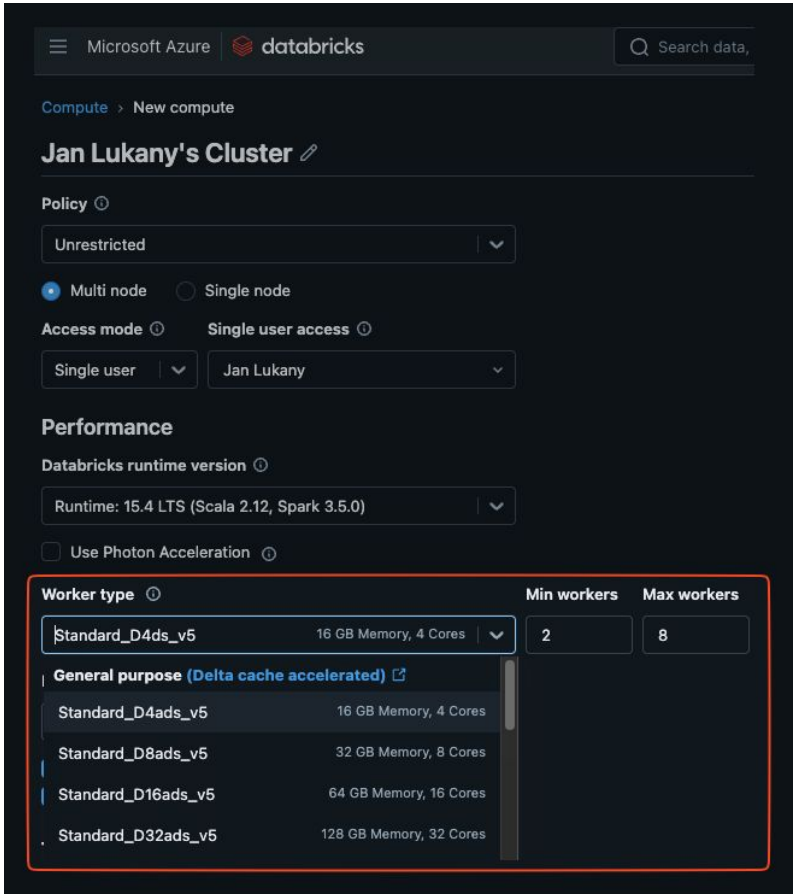
DATAMOLE

Databricks

- Compute
  - Apache Spark cluster management
  - Built-in orchestration for workflows
- Data Lakehouse
  - Delta Lake as default storage layer
  - Unity Catalog – “one” database, data governance
    - Delta tables
    - External volumes
    - Federated access
- Other
  - Development workspace (similar to Jupyter Lab) with collaborative features
  - Machine learning support (training, deployment)
  - Native integrations with major cloud providers (Azure, AWS, GCP)

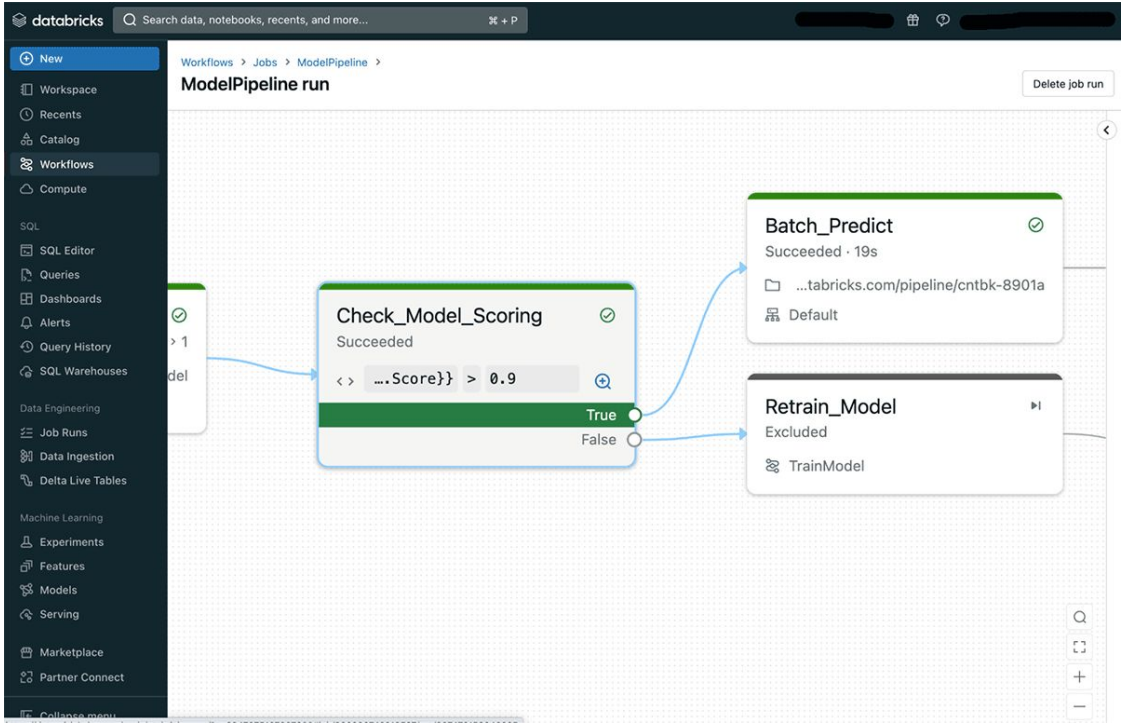
Databricks – Apache Spark cluster management

- Azure VMs
- AWS EC2
- Google Compute Engine



DATAMOLE

Databricks – Workflows (job orchestration)

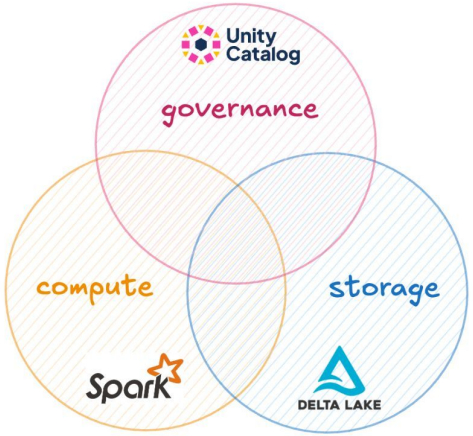


DATAMOLE

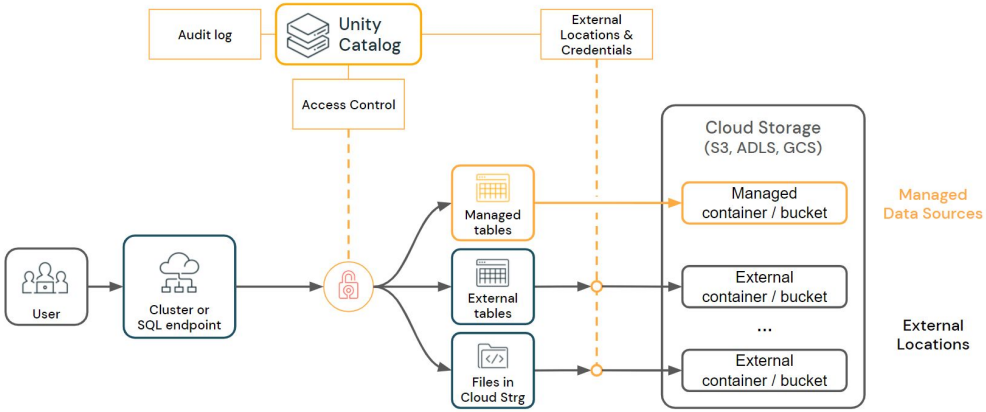
Databricks – Delta Tables on Cloud

- Amazon S3
- Azure Blob Storage
- Google Cloud Platform

Databricks – Unity Catalog



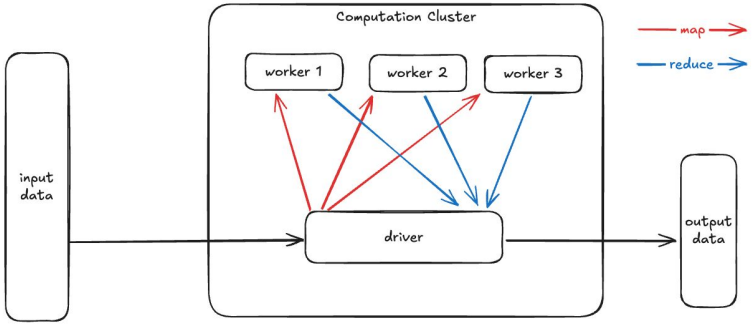
- "one" database from the perspective of consumers
- data governance at one place
- can include e.g.:
  - Delta tables
  - ML models
  - Unstructured data on cloud storages
  - External databases



DATAMOLE

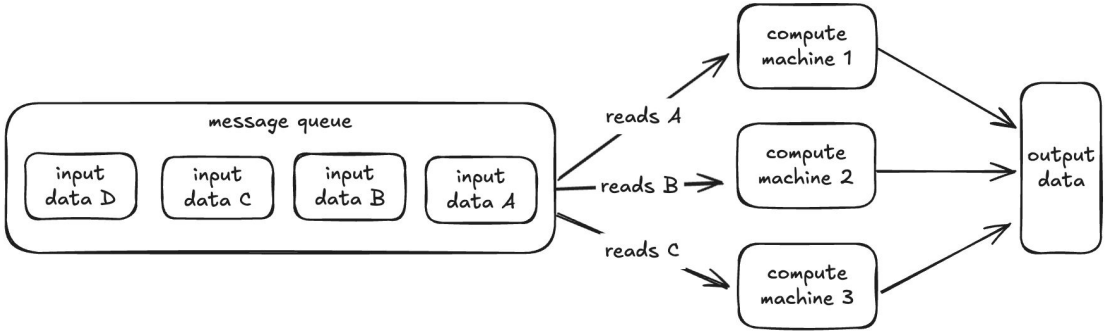
Databricks – Demo (community edition)

Map-Reduce vs Competing Consumers



Example tech stack:

- Apache Spark
- Delta Lake
- (Databricks)



Example tech stack:

- Azure Event Hub
- Kubernetes, Docker, Python
- Azure Tables (NoSQL)

DATAMOLE

Summary

- Apache Spark + Delta Lake
  - Excellent batch processing
  - Near real-time stream processing
  - Open-source
- Databricks
  - Cloud platform on top of Apache Spark + Delta Lake
  - Unity Catalog
  - ML experiments and serving
  - ...

Q&A