brightly

Rewriting tomorrow.

Future intelligent solutions,
delivered today.

Brightly introduction

Ilpo Rouhola
+358 50 487 2920
ilpo.rouhola@brightlyworks.com

Brightly in a Nutshell

Trusted strategic digital solutions partner that thinks brighter

Sectors

Process Industry	Banking & Insurance	Retail
Machinery industry	Forestry	Telecommunications
B2B services	Media	Building & Property

Key numbers

2020	90	100+
Founded	Team size	Projects

Helsinki | Amsterdam | Tampere | Oulu | Turku

Range of Services

- Architecture & Service design
- Build
- Run
- Enhance

Partners

- databricks
- aws
- Microsoft Solutions Partner — Data & AI — Azure
- snowflake
- dbt
- OPC FOUNDATION
- OPEN INDUSTRY 4.0 ALLIANCE

Ecosystem

2300+ experts in our Fusion -ecosystem

4000+ freelancers in our network

SI/Integration Partner
IoT at Scale – Demo

1. Cloud Agnostic Approach

brightly

FACTORY I/O

PLC

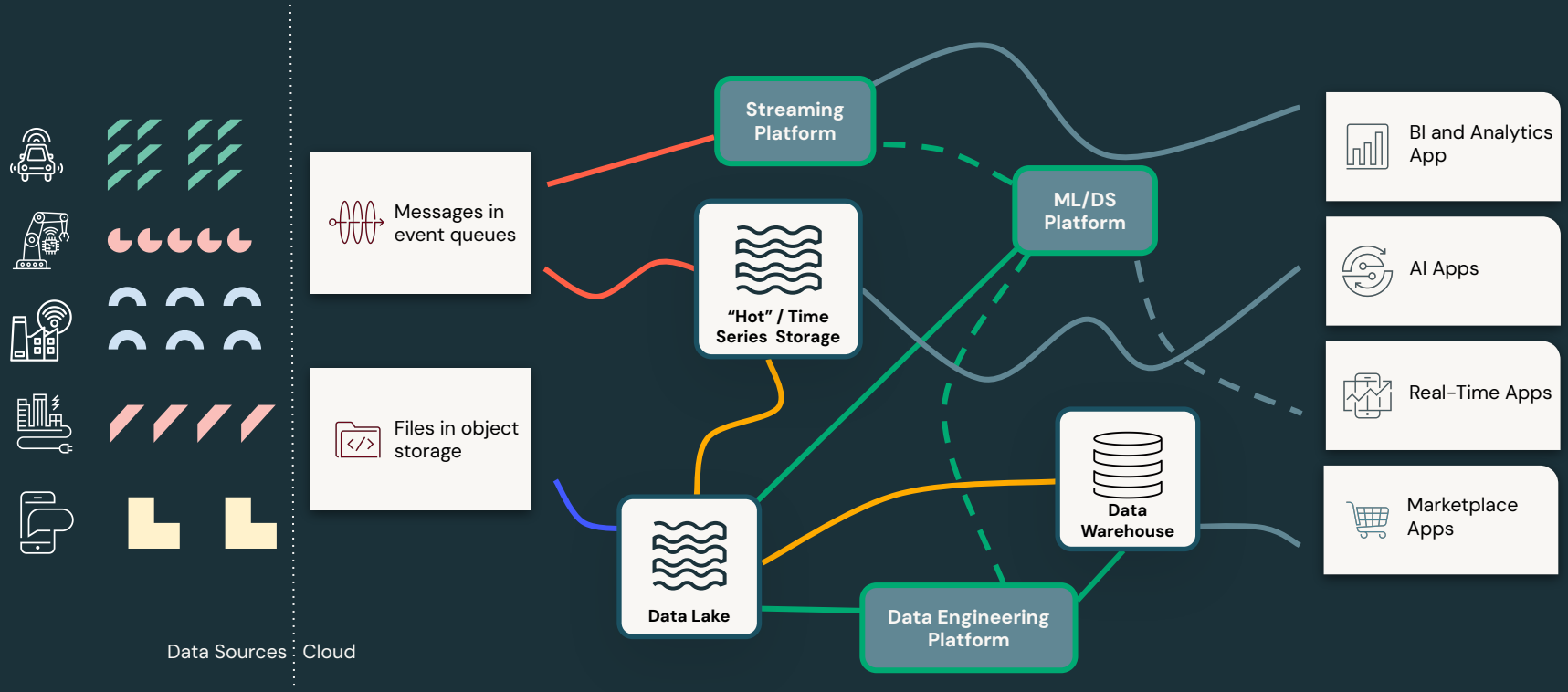
CODESYS

databricks

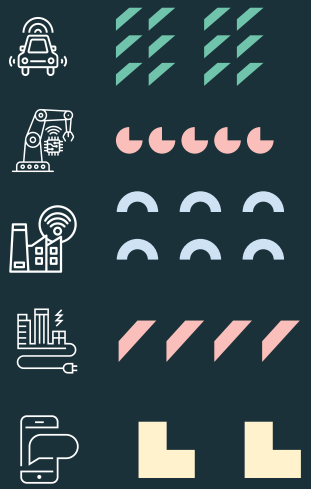
Industrial stream data management in Databricks - Jussa Klapuri

- Challenges with IOT Data
- Databricks and Lakehouse architecture
- Building Blocks
  - Spark, Delta Lake, Delta Tables
- Medallion architecture
  - Bronze, Silver, Gold
- Delta Live Tables & Data pipelines
  - Stream
  - Batch
- (Data analytics, ML/AI applications)

Today, Analytics and AI for IoT data is complicated...



...making it more difficult to deliver insights and bring AI at the right place at the right time



Data Sources

Messages in event queues

Files in object storage

Cloud

IoT data gets more silo-ed & duplicated into other data stores

Data locked in proprietary formats that require specialized skills

Stitching together many services tailored to individual use cases

Managing security & governance is more complex

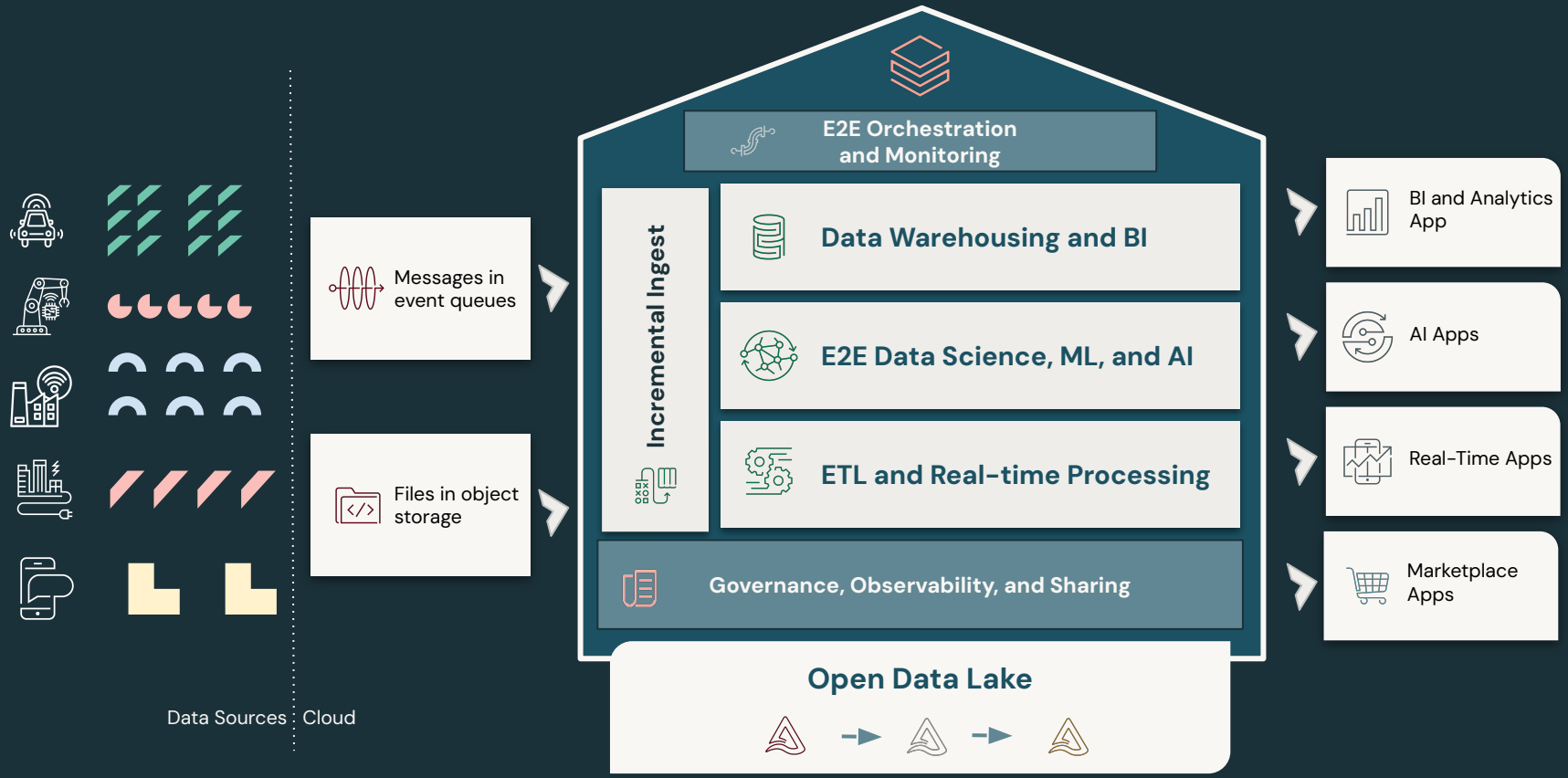
BI and Analytics App

AI Apps

Real-Time Apps

Marketplace Apps

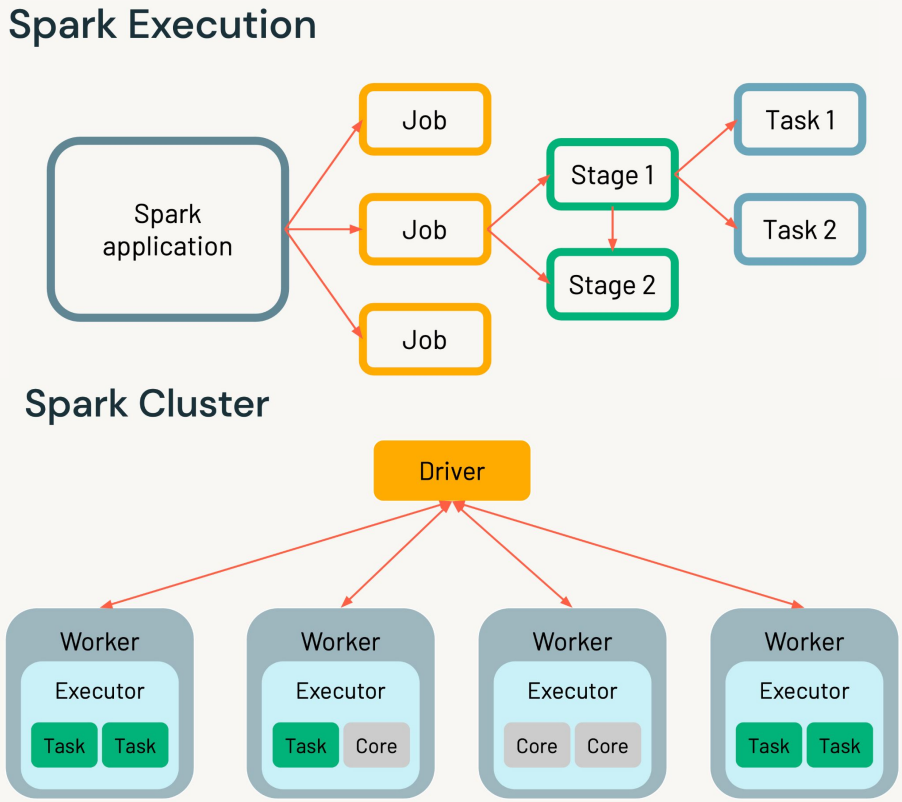
IoT Analytics and AI on the Databricks Lakehouse



Distributed computing based on Spark

- De-facto standard unified analytics engine for big data processing
- Largest open-source project in data processing
- Technology created by the founders of Databricks

Databricks as a platform makes Spark much easier and user-friendly to use



Delta Lake as Storage

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

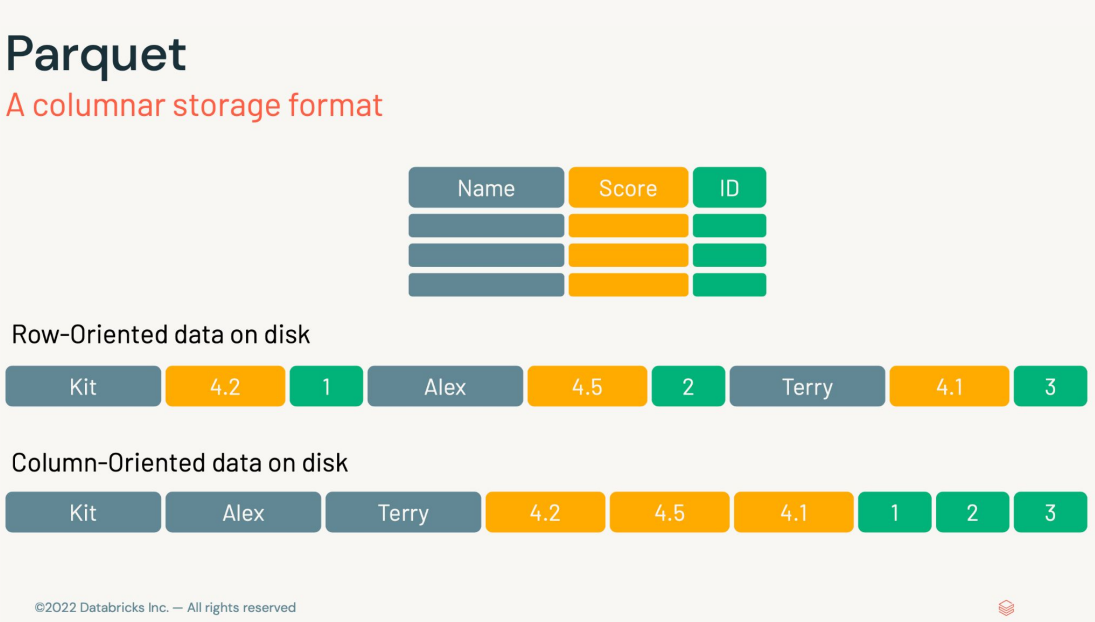
SQL, Scala/Java and Python APIs to merge, update and delete datasets

Delta tables = Parquet files + metadata (JSON)

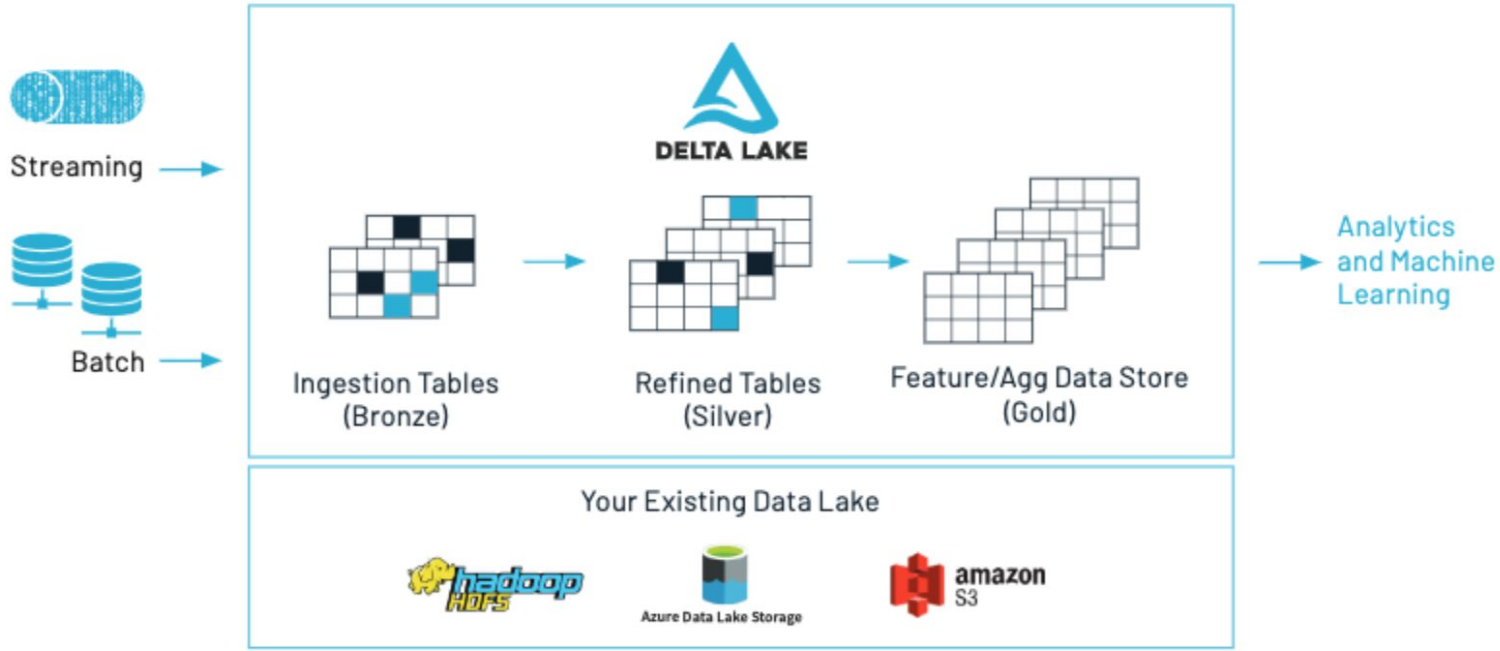
Parquet format can store data from a collection of raw OPC UA produced JSON files in 10-1000x less space

Columns contain similar data, which usually can be compressed effectively.

Supports nested data.



Medallion Architecture



Delta Live Tables

The best way to do ETL on the lakehouse

```
CREATE STREAMING TABLE raw_data
AS SELECT *
FROM cloud_files ("/raw_data",
"json")

CREATE MATERIALIZED VIEW clean_data
AS SELECT ...
FROM LIVE.raw_data
```

Accelerate ETL development

Declare **SQL or Python** and DLT automatically orchestrates the DAG, handles retries, changing data

Automatically manage your infrastructure

Automates complex tedious activities like **recovery, auto-scaling, and performance optimization**

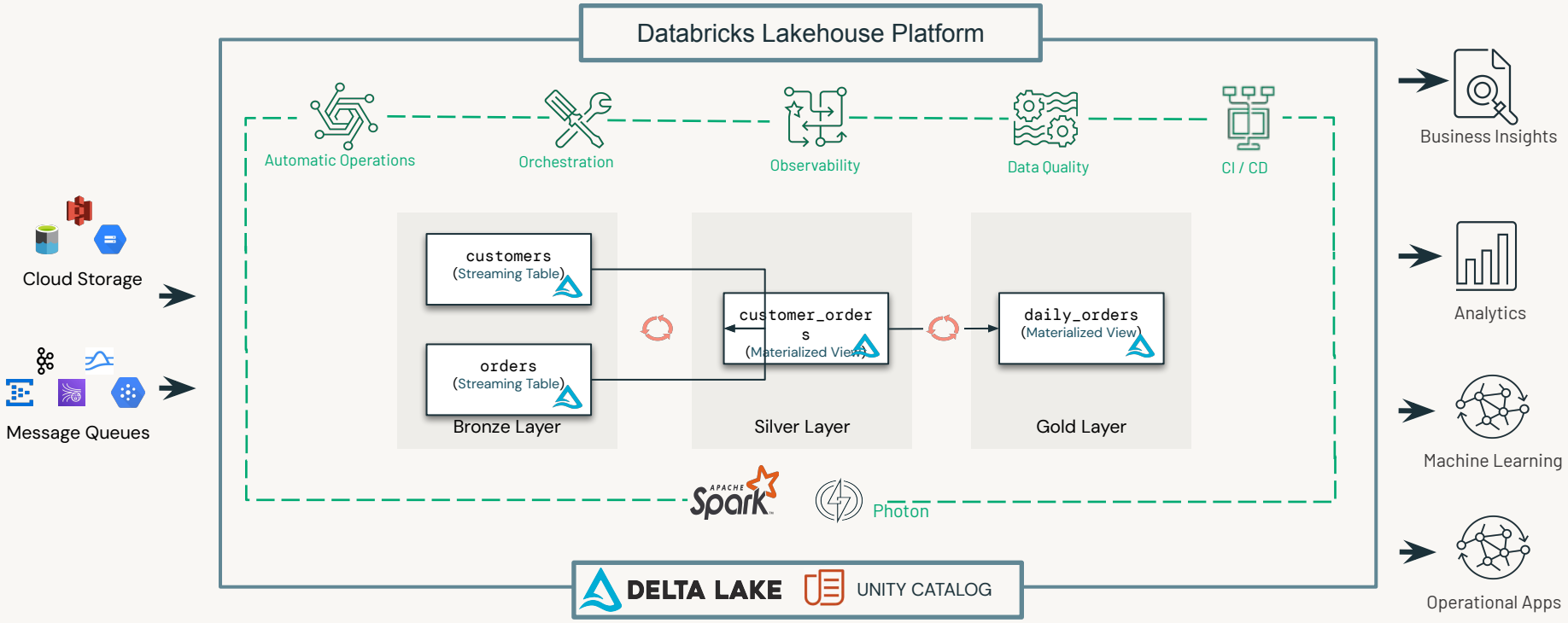
Ensure high data quality

Deliver reliable data with built-in **quality controls, testing, monitoring, and enforcement**

Unify batch and streaming

Get the simplicity of SQL with freshness of streaming with one **unified API**

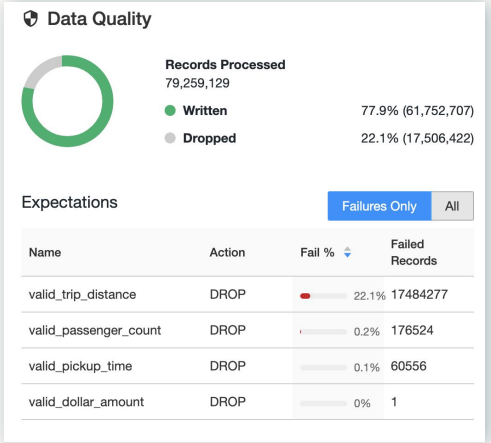
Build Production ETL Pipelines with DLT



Data quality validation and monitoring

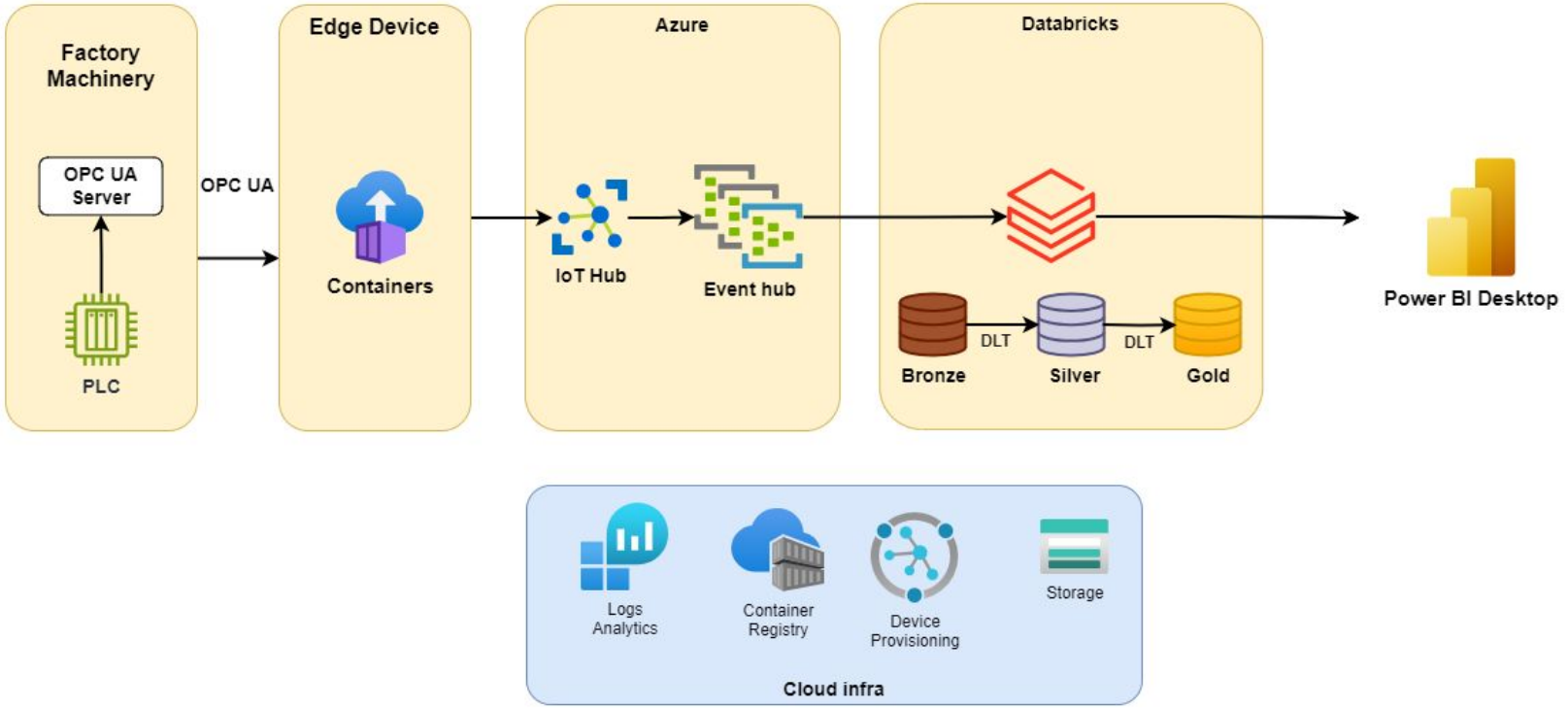
- Define data quality and integrity controls within the pipeline with data expectations
- Address data quality errors with flexible policies: fail, drop, alert, quarantine(future)
- All data pipeline runs and quality metrics are captured, tracked and reported

```
/* Stage 1: Bronze Table drop invalid rows */
CREATE STREAMING LIVE TABLE fire_account_bronze AS
( CONSTRAINT valid_account_open_dt EXPECT (acconut_dt is not null
and (account_close_dt > account_open_dt)) ON VIOLATION DROP ROW
COMMENT "Bronze table with valid account ids"
SELECT * FROM fire_account_raw ...
```

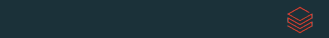


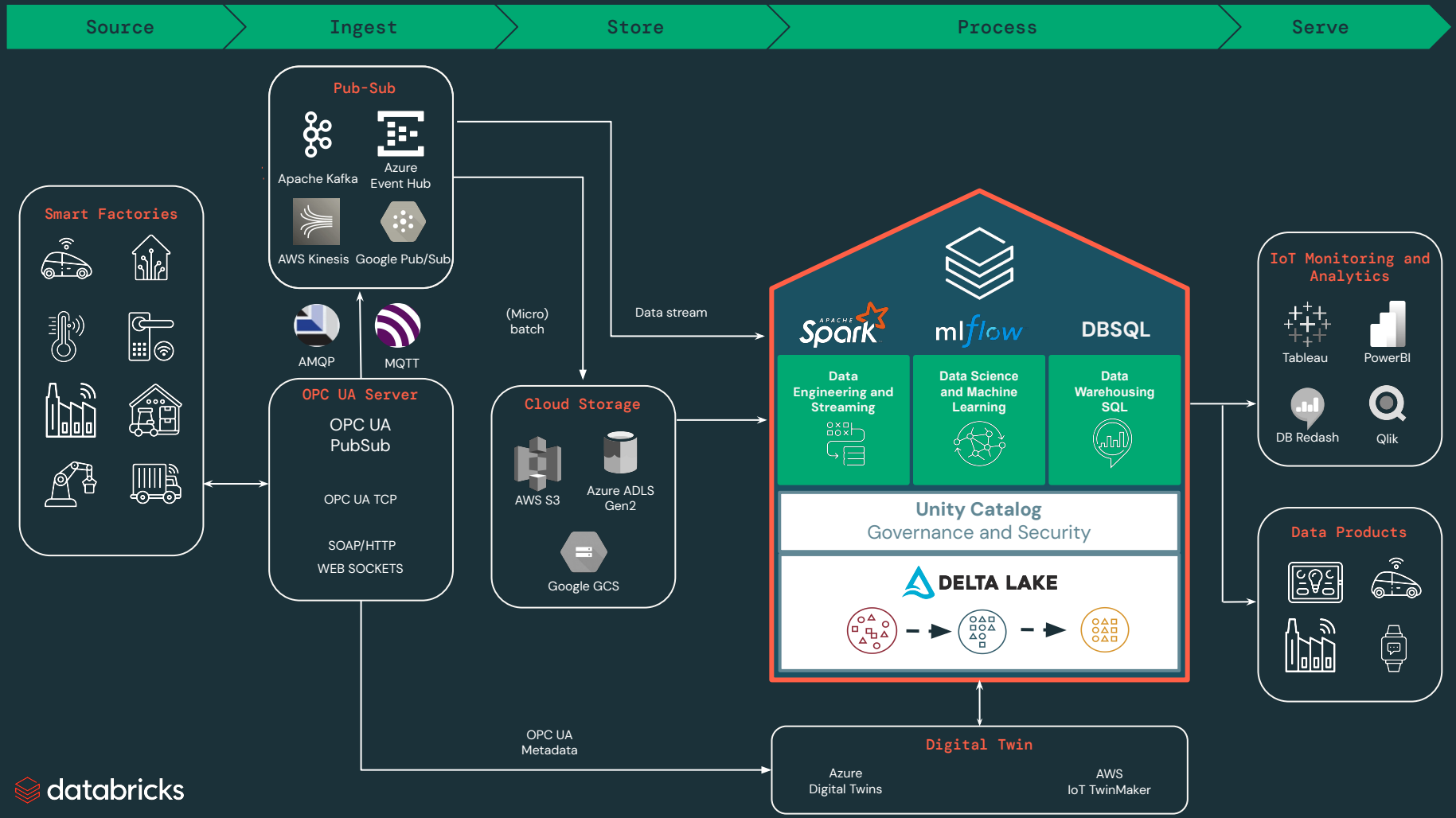
PATH TO SUCCESS

Scalable architecture for industrial data & analytics



Thank you




Source
Ingest
Store
Process
Serve
Pub-Sub
Apache Kafka
Azure Event Hub
AWS Kinesis
Google Pub/Sub
Smart Factories
AMQP
MQTT
(Micro) batch
Data stream
OPC UA Server
OPC UA PubSub
OPC UA TCP
SOAP/HTTP
WEB SOCKETS
Cloud Storage
AWS S3
Azure ADLS Gen2
Google GCS
Spark
mlflow
DBSQL
Data Engineering and Streaming
Data Science and Machine Learning
Data Warehousing SQL
Unity Catalog
Governance and Security
DELTA LAKE
IoT Monitoring and Analytics
Tableau
PowerBI
DB Redash
Qlik
Data Products
OPC UA Metadata
Digital Twin
Azure Digital Twins
AWS IoT TwinMaker
databricks

Lakehouse architecture, transformations

FILE STORAGE

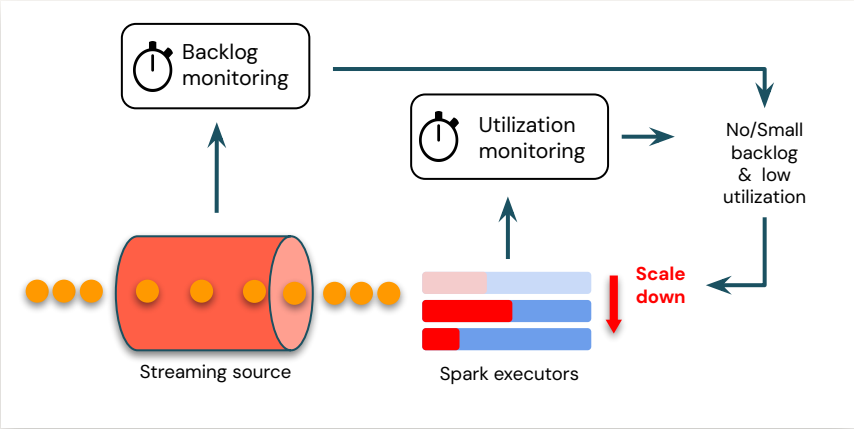
BRONZE	SILVER	GOLD		
		FACT TABLE/VIEW GOLD	DIMENSION TABLE/VIEW GOLD	FLAT TABLE/VIEW GOLD
DATASOURCE/ DATASET/ YYYY/ MM/ DD/ ANY FILE FORMAT FILE	“SILVER_DATABASE” /ERP/ “TABLE”/ YYYY/ MM/ DD/ (OR CUSTOM PARTITIONING) PARQUET/DELTA FILE	“GOLD_DATABASE”/ “F_TABLE”/ CUSTOM PARTITIONING/ PARQUET/DELTA FILE	“GOLD_DATABASE”/ “DIM_TABLE” PARQUET/DELTA FILE	“GOLD_DATABASE”/ “TABLE”/ CUSTOM PARTITIONING(IF NEEDED/ PARQUET/DELTA FILE

Enhanced Autoscaling

Save infrastructure costs while maintaining end-to-end latency SLAs for streaming workloads

Problem

Optimize infrastructure spend when making scaling decisions for streaming workloads



- Built to handle streaming workloads which are spiky and unpredictable
- Shuts down nodes when utilization is low while guaranteeing task execution
- Only scales up to needed # of nodes

AWS	Azure	GCP
Generally Available	Generally Available	Public Preview GA Coming Soon