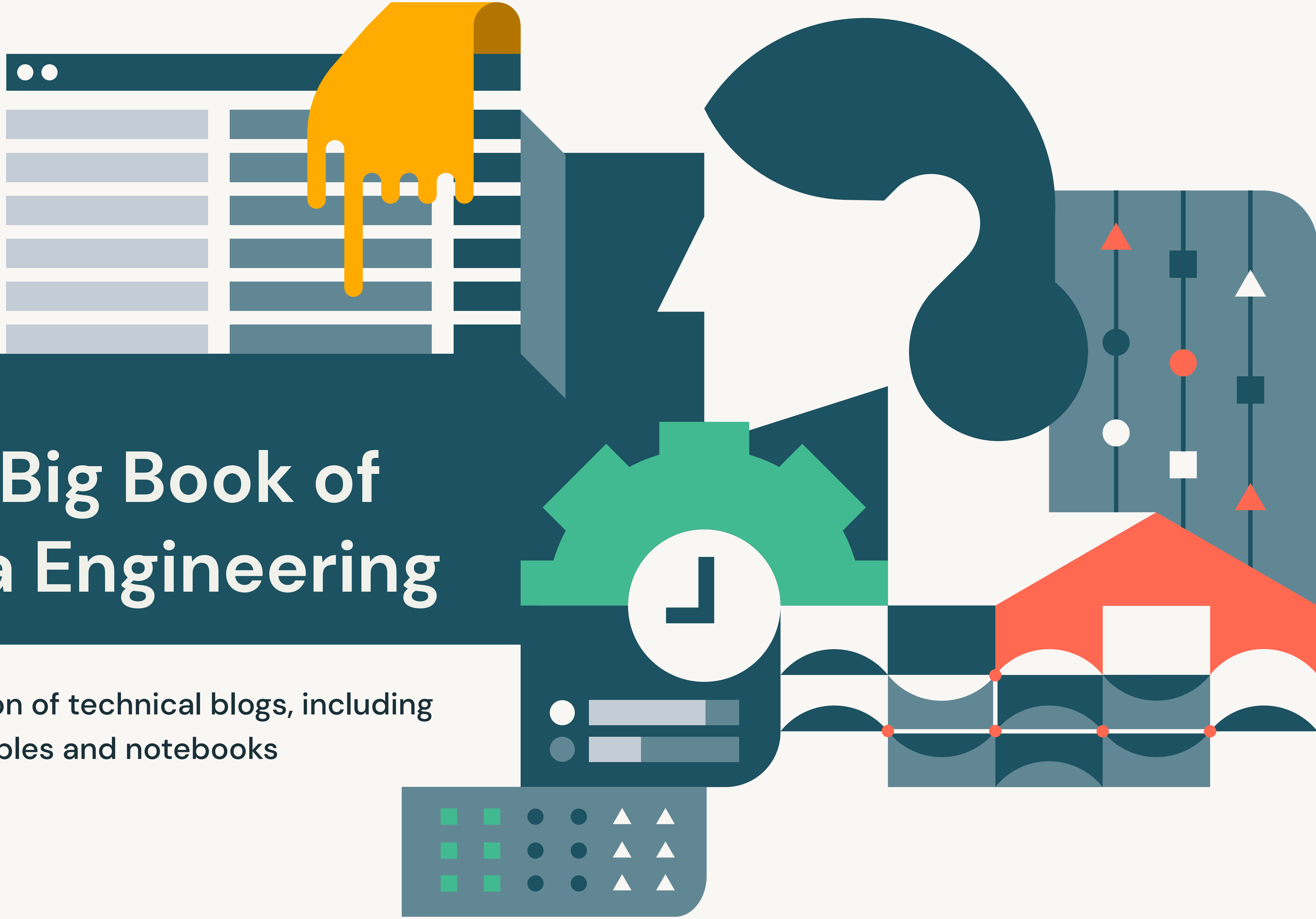
EBOOK

The Big Book of Data Engineering

A collection of technical blogs, including code samples and notebooks



databricks

Contents

SECTION

01

Introduction to Data Engineering on Databricks

Organizations realize the value data plays as a strategic asset for various business-related initiatives, such as growing revenues, improving the customer experience, operating efficiently or improving a product or service. However, accessing and managing data for these initiatives has become increasingly complex. Most of the complexity has arisen with the explosion of data volumes and data types, with organizations amassing an estimated 80% of data in unstructured and semi-structured format. As the collection of data continues to increase, 73% of the data goes unused for analytics or decision-making. In order to try and decrease this percentage and make more data usable, data engineering teams are responsible for building data pipelines to efficiently and reliably deliver data. But the process of building these complex data pipelines comes with a number of difficulties:

- In order to get data into a data lake, data engineers are required to spend immense time hand-coding repetitive data ingestion tasks
- Since data platforms continuously change, data engineers spend time building and maintaining, and then rebuilding, complex scalable infrastructure
- With the increasing importance of real-time data, low latency data pipelines are required, which are even more difficult to build and maintain
- Finally, with all pipelines written, data engineers need to constantly focus on performance, tuning pipelines and architectures to meet SLAs

How can Databricks help?

With the Databricks Lakehouse Platform, data engineers have access to an end-to-end data engineering solution for ingesting, transforming, processing, scheduling and delivering data. The Lakehouse Platform automates the complexity of building and maintaining pipelines and running ETL workloads directly on a data lake so data engineers can focus on quality and reliability to drive valuable insights.

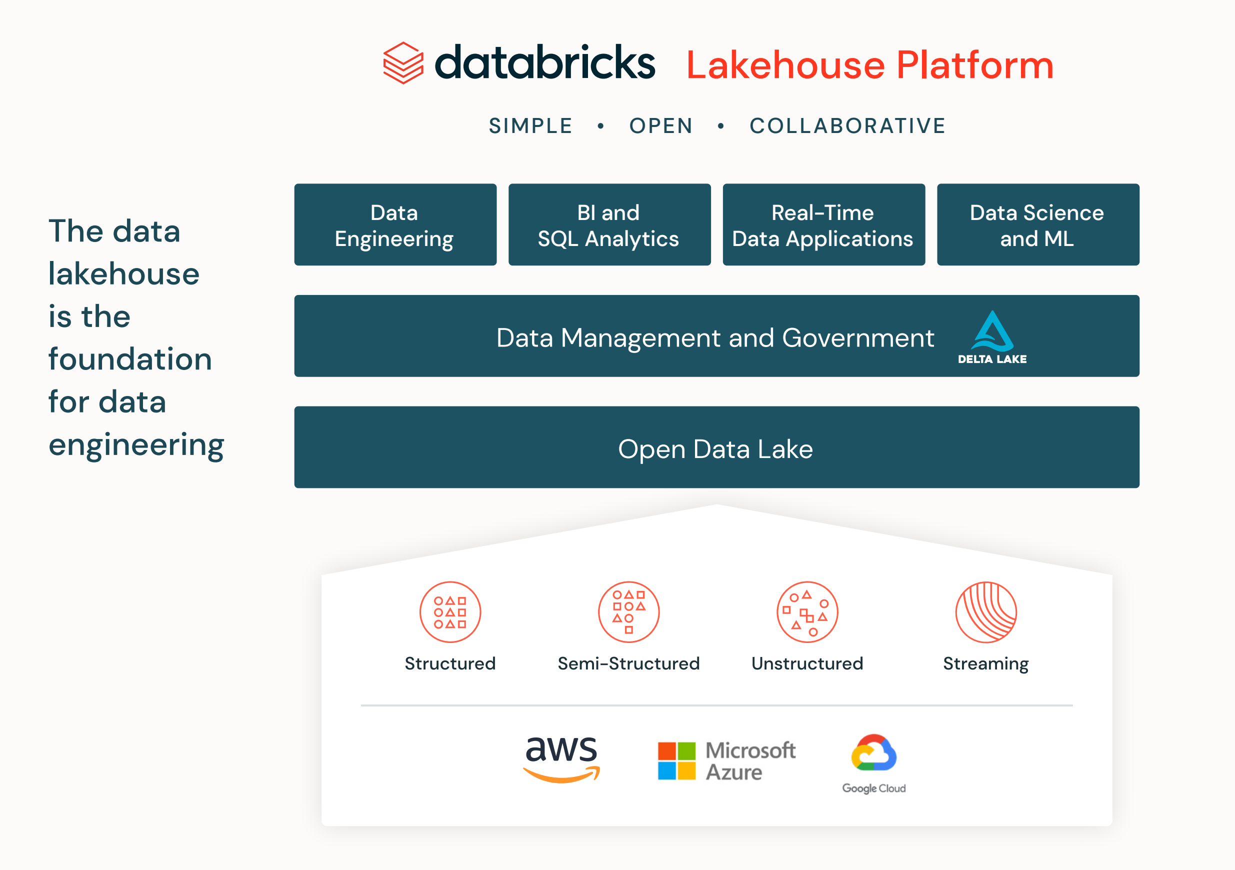

databricks Lakehouse Platform
SIMPLE • OPEN • COLLABORATIVE
The data lakehouse is the foundation for data engineering
Data Engineering
BI and SQL Analytics
Real-Time Data Applications
Data Science and ML
Data Management and Government
DELTA LAKE
Open Data Lake
Structured
Semi-Structured
Unstructured
Streaming
aws
Microsoft Azure
Google Cloud

Figure 1
The Databricks Lakehouse Platform unifies your data, analytics and AI on one common platform for all your data use cases

Key differentiators for successful data engineering with Databricks

By simplifying on a lakehouse architecture, data engineers need an enterprise-grade and enterprise-ready approach to building data pipelines. To be successful, a data engineering solution team must embrace these eight key differentiating capabilities:

Continuous or scheduled data ingestion

With the ability to ingest petabytes of data with auto-evolving schemas, data engineers can deliver fast, reliable, scalable and automatic data for analytics, data science or machine learning. This includes:

- Incrementally and efficiently processing data as it arrives from files or streaming sources like Kafka, DBMS and NoSQL
- Automatically inferring schema and detecting column changes for structured and unstructured data formats
- Automatically and efficiently tracking data as it arrives with no manual intervention
- Preventing data loss by rescuing data columns

Declarative ETL pipelines

Data engineers can reduce development time and effort and instead focus on implementing business logic and data quality checks within the data pipeline using SQL or Python. This can be achieved by:

- Using intent-driven declarative development to simplify "how" and define "what" to solve
- Automatically creating high-quality lineage and managing table dependencies across the data pipeline
- Automatically checking for missing dependencies or syntax errors, and managing data pipeline recovery

Data quality validation and monitoring

Improve data reliability throughout the data lakehouse so data teams can confidently trust the information for downstream initiatives by:

- Defining data quality and integrity controls within the pipeline with defined data expectations
- Addressing data quality errors with predefined policies (fail, drop, alert, quarantine)
- Leveraging the data quality metrics that are captured, tracked and reported for the entire data pipeline

Fault tolerant and automatic recovery

Handle transient errors and recover from most common error conditions occurring during the operation of a pipeline with fast, scalable automatic recovery that includes:

- Fault tolerant mechanisms to consistently recover the state of data
- The ability to automatically track progress from the source with checkpointing
- The ability to automatically recover and restore the data pipeline state

Data pipeline observability

Monitor overall data pipeline status from a dataflow graph dashboard and visually track end-to-end pipeline health for performance, quality and latency. Data pipeline observability capabilities include:

- A high-quality, high-fidelity lineage diagram that provides visibility into how data flows for impact analysis
- Granular logging with performance and status of the data pipeline at a row level
- Continuous monitoring of data pipeline jobs to ensure continued operation

Batch and stream data processing

Allow data engineers to tune data latency with cost controls without the need to know complex stream processing or implement recovery logic.

- Execute data pipeline workloads on automatically provisioned elastic Apache Spark™-based compute clusters for scale and performance
- Use performance optimization clusters that parallelize jobs and minimize data movement

Automatic deployments and operations

Ensure reliable and predictable delivery of data for analytics and machine learning use cases by enabling easy and automatic data pipeline deployments and rollbacks to minimize downtime. Benefits include:

- Complete, parameterized and automated deployment for the continuous delivery of data
- End-to-end orchestration, testing and monitoring of data pipeline deployment across all major cloud providers

Scheduled pipelines and workflows

Simple, clear and reliable orchestration of data processing tasks for data and machine learning pipelines with the ability to run multiple non-interactive tasks as a directed acyclic graph (DAG) on a Databricks compute cluster.

- Easily orchestrate tasks in a DAG using the Databricks UI and API
- Create and manage multiple tasks in jobs via UI or API and features, such as email alerts for monitoring
- Orchestrate any task that has an API outside of Databricks and across all clouds

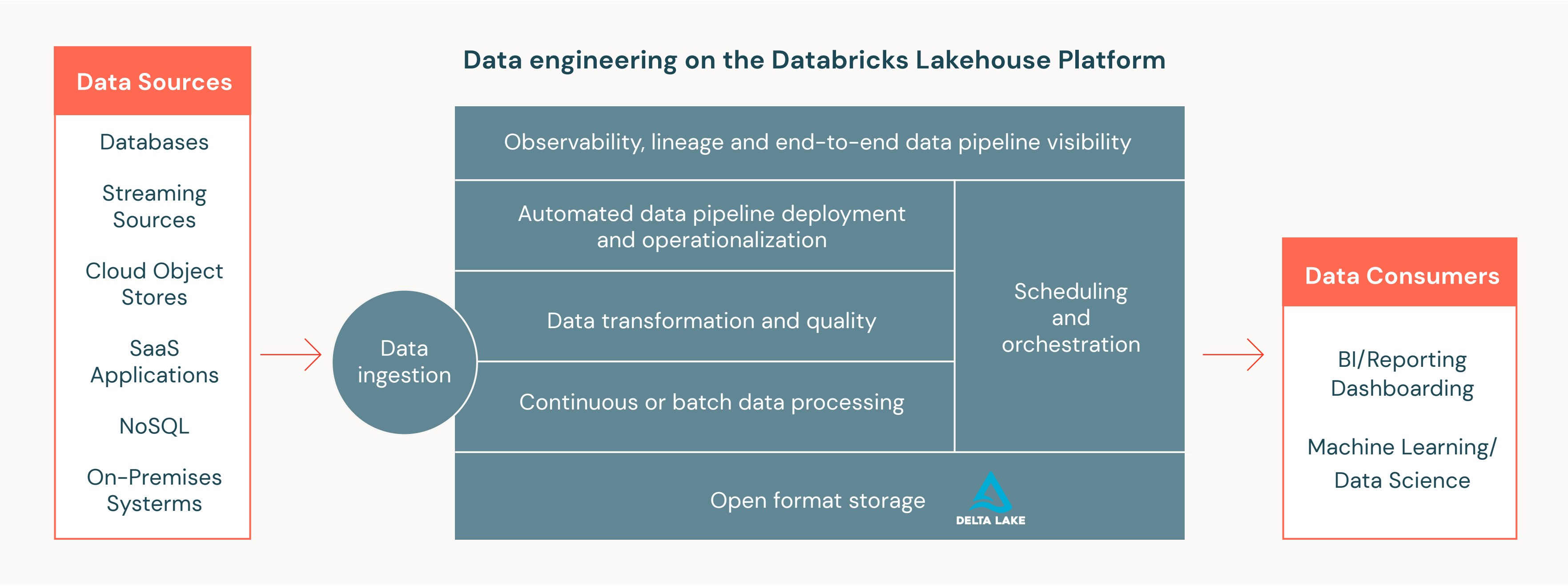

Data engineering on the Databricks Lakehouse Platform
Data Sources
Databases
Streaming Sources
Cloud Object Stores
SaaS Applications
NoSQL
On-Premises Systerms
Data ingestion
Observability, lineage and end-to-end data pipeline visibility
Automated data pipeline deployment and operationalization
Data transformation and quality
Continuous or batch data processing
Scheduling and orchestration
Open format storage
DELTA LAKE
Data Consumers
BI/Reporting Dashboarding
Machine Learning/ Data Science

Figure 2
Data engineering on Databricks reference architecture

Conclusion

As organizations strive to become data-driven, data engineering is a focal point for success. To deliver reliable, trustworthy data, data engineers shouldn't need to spend time manually developing and maintaining an end-to-end ETL lifecycle. Data engineering teams need an efficient, scalable way to simplify ETL development, improve data reliability and manage operations.

As described, the eight key differentiating capabilities simplify the management of the ETL lifecycle by automating and maintaining all data dependencies, leveraging built-in quality controls with monitoring and providing deep visibility into pipeline operations with automatic recovery. Data engineering teams can now focus on easily and rapidly building reliable end-to-end production-ready data pipelines using only SQL or Python for batch and streaming that delivers high-value data for analytics, data science or machine learning.

Use cases

In the next section, we describe best practices for data engineering end-to-end use cases drawn from real-world examples. From data ingestion and data processing to analytics and machine learning, you'll learn how to translate raw data into actionable data. We'll arm you with the data sets and code samples, so you can get your hands dirty as you explore all aspects of the data lifecycle on the Databricks Lakehouse Platform.

SECTION

02

Real-Life Use Cases on the Databricks Lakehouse Platform

SECTION 2.1

Real-Time Point-of-Sale Analytics With the Data Lakehouse

by BRYAN SMITH and ROB SAKER

September 9, 2021

Disruptions in the supply chain — from reduced product supply and diminished warehouse capacity — coupled with rapidly shifting consumer expectations for seamless omnichannel experiences are driving retailers to rethink how they use data to manage their operations. Prior to the pandemic, 71% of retailers named lack of real-time visibility into inventory as a top obstacle to achieving their omnichannel goals. The pandemic only increased demand for integrated online and in-store experiences, placing even more pressure on retailers to present accurate product availability and manage order changes on the fly. Better access to real-time information is the key to meeting consumer demands in the new normal.

In this blog, we'll address the need for real-time data in retail, and how to overcome the challenges of moving real-time streaming of point-of-sale data at scale with a data lakehouse.

The point-of-sale system

The point-of-sale (POS) system has long been the central piece of in-store infrastructure, recording the exchange of goods and services between retailer and customer. To sustain this exchange, the POS typically tracks product inventories and facilitates replenishment as unit counts dip below critical levels. The importance of the POS to in-store operations cannot be overstated, and as the system of record for sales and inventory operations, access to its data is of key interest to business analysts.

Historically, limited connectivity between individual stores and corporate offices meant the POS system (not just its terminal interfaces) physically resided within the store. During off-peak hours, these systems might phone home to transmit summary data, which when consolidated in a data warehouse, provide a day-old view of retail operations performance that grows increasingly stale until the start of the next night's cycle.

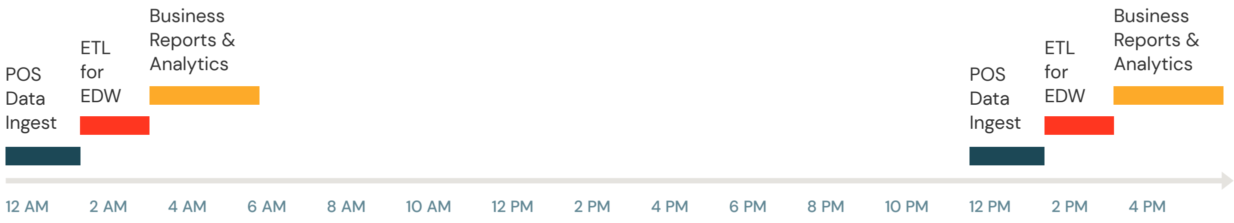


Figure 1
Inventory availability with traditional, batch-oriented ETL patterns

Modern connectivity improvements have enabled more retailers to move to a centralized, cloud-based POS system, while many others are developing near real-time integrations between in-store systems and the corporate back office. Near real-time availability of information means that retailers can continuously update their estimates of item availability. No longer is the business managing operations against their knowledge of inventory states as they were a day prior but instead is taking actions based on their knowledge of inventory states as they are now.

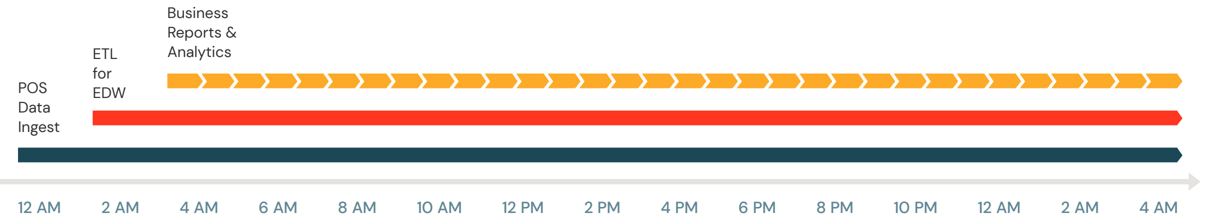


Figure 2
Inventory availability with streaming ETL patterns

Near real-time insights

As impactful as near real-time insights into store activity are, the transition from nightly processes to continuous streaming of information brings particular challenges, not only for the data engineer, who must design a different kind of data processing workflow, but also for the information consumer. In this post, we share some lessons learned from customers who've recently embarked on this journey and examine how key patterns and capabilities available through the lakehouse pattern can enable success.

LESSON 1
Carefully consider scope

POS systems are often not limited to just sales and inventory management. Instead, they can provide a sprawling range of functionality, including payment processing, store credit management, billing and order placement, loyalty program management, employee scheduling, time-tracking and even payroll, making them a veritable Swiss Army knife of in-store functionality.

As a result, the data housed within the POS is typically spread across a large and complex database structure. If lucky, the POS solution makes a data access layer available, which makes this data accessible through more easily interpreted structures. But if not, the data engineer must sort through what can be an opaque set of tables to determine what is valuable and what is not.

Regardless of how the data is exposed, the classic guidance holds true: identify a compelling business justification for your solution and use that to limit the scope of the information assets you initially consume. Such a justification often comes from a strong business sponsor, who is tasked with addressing a specific business challenge and sees the availability of more timely information as critical to their success.

To illustrate this, consider a key challenge for many retail organizations today: the enablement of omnichannel solutions. Such solutions, which enable buy-online, pickup in-store (BOPIS) and cross-store transactions, depend on reasonably accurate information about store inventory. If we were to limit our initial scope to this one need, the information requirements for our monitoring and analytics system become dramatically reduced. Once a real-time inventory solution is delivered and value is recognized by the business, we can expand our scope to consider other needs, such as promotions monitoring and fraud detection, expanding the breadth of information assets leveraged with each iteration.

LESSON 2

Align transmission with patterns of data generation and time sensitivities

Different processes generate data differently within the POS. Sales transactions are likely to leave a trail of new records appended to relevant tables. Returns may follow multiple paths, triggering updates to past sales records, the insertion of new, reversing sales records and/or the insertion of new information in returns-specific structures. Vendor documentation, tribal knowledge and even some independent investigative work may be required to uncover exactly how and where event-specific information lands within the POS.

Understanding these patterns can help build a data transmission strategy for specific kinds of information. Higher frequency, finer-grained, insert-oriented patterns may be ideally suited for continuous streaming. Less frequent, larger-scale events may best align with batch-oriented, bulk data styles of transmission. But if these modes of data transmission represent two ends of a spectrum, you are likely to find most events captured by the POS fall somewhere in between.

The beauty of the data lakehouse approach to data architecture is that multiple modes of data transmission can be employed in parallel. For data naturally aligned with the continuous transmission, streaming may be employed. For data better aligned with bulk transmission, batch processes may be used. And for those data falling in the middle, you can focus on the timeliness of the data required for decision-making and allow that to dictate the path forward. All of these modes can be tackled with a consistent approach to ETL implementation, a challenge that thwarted many earlier implementations of what were frequently referred to as lambda architectures.

LESSON 3

Land the data in stages

Data arrives from the in-store POS systems with different frequencies, formats and expectations for timely availability. Leveraging the Bronze, Silver & Gold design pattern popular within lakehouses, you can separate initial cleansing, reformatting and persistence of the data from the more complex transformations required for specific business-aligned deliverables.

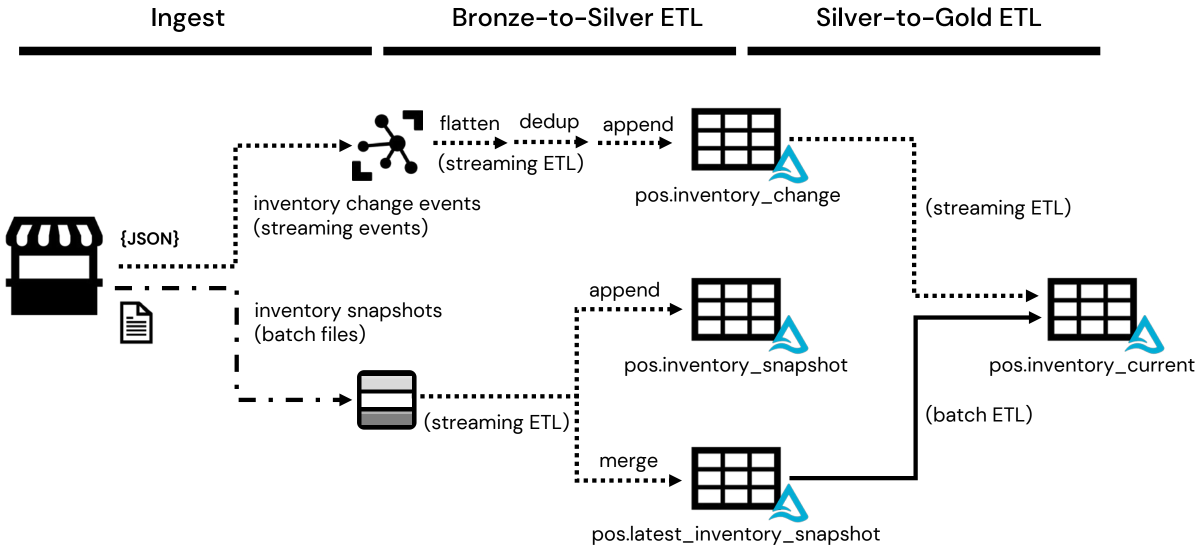


Figure 3
A data lakehouse architecture for the calculation of current inventory leveraging the Bronze, Silver and Gold pattern of data persistence

LESSON 4

Manage expectations

The move to near real-time analytics requires an organizational shift. Gartner describes this through their Streaming Analytics Maturity model within which analysis of streaming data becomes integrated into the fabric of day-to-day operations. This does not happen overnight.

Instead, data engineers need time to recognize the challenges inherent to streaming delivery from physical store locations into a centralized, cloud-based back office. Improvements in connectivity and system reliability coupled with increasingly robust ETL workflows land data with greater timeliness, reliability and consistency. This often entails enhancing partnerships with systems engineers and application developers to support a level of integration not typically present in the days of batch-only ETL workflows.

Business analysts will need to become familiar with the inherent noisiness of data being updated continuously. They will need to relearn how to perform diagnostic and validation work on a data set, such as when a query that ran seconds prior now returns a slightly different result. They must gain a deeper awareness of the problems in the data which are often hidden when presented in daily aggregates. All of this will require adjustments both to their analysis and their response to detected signals in their results.

All of this takes place in just the first few stages of maturation. In later stages, the organization's ability to detect meaningful signals within the stream may lead to more automated sense and response capabilities. Here, the highest levels of value in the data streams are unlocked. But monitoring and governance must be put into place and proven before the business will entrust its operations to these technologies.

Implementing POS streaming

To illustrate how the lakehouse architecture can be applied to POS data, we've developed a demonstration workflow within which we calculate a near real-time inventory. In it, we envision two separate POS systems transmitting inventory-relevant information associated with sales, restocks and shrinkage data along with buy-online, pickup in-store (BOPIS) transactions (initiated in one system and fulfilled in the other) as part of a streaming inventory change feed. Periodic (snapshot) counts of product units on-shelf are captured by the POS and transmitted in bulk. These data are simulated for a one-month period and played back at 10x speed for greater visibility into inventory changes.

The ETL processes (as pictured in Figure 3) represent a mixture of streaming and batch techniques. A two-staged approach with minimally transformed data captured in Delta tables representing our Silver layer separates our initial, more technically aligned ETL approach with the more business-aligned approach required for current inventory calculations. The second stage has been implemented using traditional structured streaming capabilities, something we may revisit with the new Delta Live Tables functionality as it makes its way toward general availability.

The demonstration makes use of Azure IOT Hubs and Azure Storage for data ingestion but would work similarly on the AWS and GCP clouds with appropriate technology substitutions.

Start experimenting with these free Databricks notebooks

- **POS 01: Environment Setup**
- **POS 02: Data Generation**
- **POS 03: Ingest ETL**
- **POS 04: Current Inventory**

SECTION 2.2

Building a Cybersecurity Lakehouse for CrowdStrike Falcon Events

by AEMRO AMARE, ARUN PAMULAPATI, YONG SHENG HUANG and JASON POHL

May 20, 2021

Endpoint data is required by security teams for threat detection, threat hunting, incident investigations and to meet compliance requirements. The data volumes can be terabytes per day or petabytes per year. Most organizations struggle to collect, store and analyze endpoint logs because of the costs and complexities associated with such large data volumes. But it doesn't have to be this way.

In this two-part blog series, we will cover how you can operationalize petabytes of endpoint data with Databricks to improve your security posture with advanced analytics in a cost-effective way. Part 1 (this blog) will cover the architecture of data collection and the integration with a SIEM (Splunk). At the end of this blog, with notebooks provided, you will be ready to use the data for analysis. Part 2 will discuss specific use cases, how to create ML models and automated enrichments and analytics. At the end of part 2, you will be able to implement the notebooks to detect and investigate threats using endpoint data.

We will use CrowdStrike's Falcon logs as our example. To access Falcon logs, one can use the Falcon Data Replicator (FDR) to push raw event data from CrowdStrike's platform to cloud storage such as Amazon S3. This data can be ingested, transformed, analyzed and stored using the Databricks Lakehouse Platform alongside the rest of their security telemetry. Customers can ingest CrowdStrike Falcon data, apply Python-based real-time detections, search through historical data with Databricks SQL, and query from SIEM tools like Splunk with Databricks Add-on for Splunk.

Challenge of operationalizing CrowdStrike data

Although the CrowdStrike Falcon data offers comprehensive event logging details, it is a daunting task to ingest, process and operationalize complex and large volumes of cybersecurity data on a near real-time basis in a cost-effective manner. These are a few of the well-known challenges:

- **Real-time data ingestion at scale:** It is difficult to keep track of processed and unprocessed raw data files, which are written by FDR on cloud storage in near real time.
- **Complex transformations:** The data format is semi-structured. Every line of each log file contains hundreds of underministically different types of payloads, and the structure of event data can change over time.
- **Data governance:** This kind of data can be sensitive, and access must be gated to only users who need it.

- **Simplified security analytics end-to-end:** Scalable tools are needed to do the data engineering, ML and analysis on these fast-moving and high-volume data sets.
- **Collaboration:** Effective collaboration can leverage domain expertise from the data engineers, cybersecurity analysts and ML engineers. Thus, having a collaborative platform improves the efficiency of cybersecurity analysis and response workloads.

As a result, security engineers across enterprises find themselves in a difficult situation, struggling to manage cost and operational efficiency. They either have to accept being locked into very expensive proprietary systems or spend tremendous effort to build their own endpoint security tools while fighting for scalability and performance.

Databricks cybersecurity lakehouse

Databricks offers security teams and data scientists a new hope to perform their jobs efficiently and effectively, as well as a set of tools to combat the growing challenges of big data and sophisticated threats.

Lakehouse, an open architecture that combines the best elements of data lakes and data warehouses, simplifies building a multi-hop data engineering pipeline that progressively adds structure to the data. The benefit of a multi-hop architecture is that data engineers can build a pipeline that begins with raw data as a "single source of truth" from which everything flows. CrowdStrike's semi-structured raw data can be stored for years, and subsequent transformations and aggregations can be done in an end-to-end streaming fashion to refine the data and introduce context-specific structure to analyze and detect security risks in different scenarios.

- **Data ingestion:** Auto Loader (AWS | Azure | GCP) helps to immediately read data as soon as a new file is written by CrowdStrike FDR into raw data storage. It leverages cloud notification services to incrementally process new files as they arrive on the cloud. Auto Loader also automatically configures and listens to the notification service for new files and can scale up to millions of files per second.
- **Unified stream and batch processing:** Delta Lake is an open approach to bringing data management and governance to data lakes that leverages the distributed computation power of Apache Spark™ for huge volumes of data and metadata. Databricks Delta Engine is a highly optimized engine that can process millions of records per second.
- **Data governance:** With Databricks Table Access Control (AWS | Azure | GCP), admins can grant different levels of access to Delta tables based on a user's business function.

- **Security analysis tools:** Databricks SQL helps to create an interactive dashboard with automatic alerting when unusual patterns are detected. Likewise, it can easily integrate with highly adopted BI tools such as Tableau, Microsoft Power BI and Looker.
- **Collaboration on Databricks notebooks:** Databricks collaborative notebooks enable security teams to collaborate in real time. Multiple users can run queries in multiple languages, share visualizations and make comments within the same workspace to keep investigations moving forward without interruption.

Lakehouse architecture for CrowdStrike Falcon data

We recommend the following lakehouse architecture for cybersecurity workloads, such as CrowdStrike's Falcon data. Auto Loader and Delta Lake simplify the process of reading raw data from cloud storage and writing to a Delta table at low cost and with minimal DevOps work.

In this architecture, semi-structured CrowdStrike data is loaded to the customer's cloud storage in the landing zone. Then Auto Loader uses cloud notification services to automatically trigger the processing and ingestion of new files into the customer's Bronze tables, which will act as the single source of truth for all downstream jobs. Auto Loader will track processed and unprocessed files using checkpoints in order to prevent duplicate data processing.

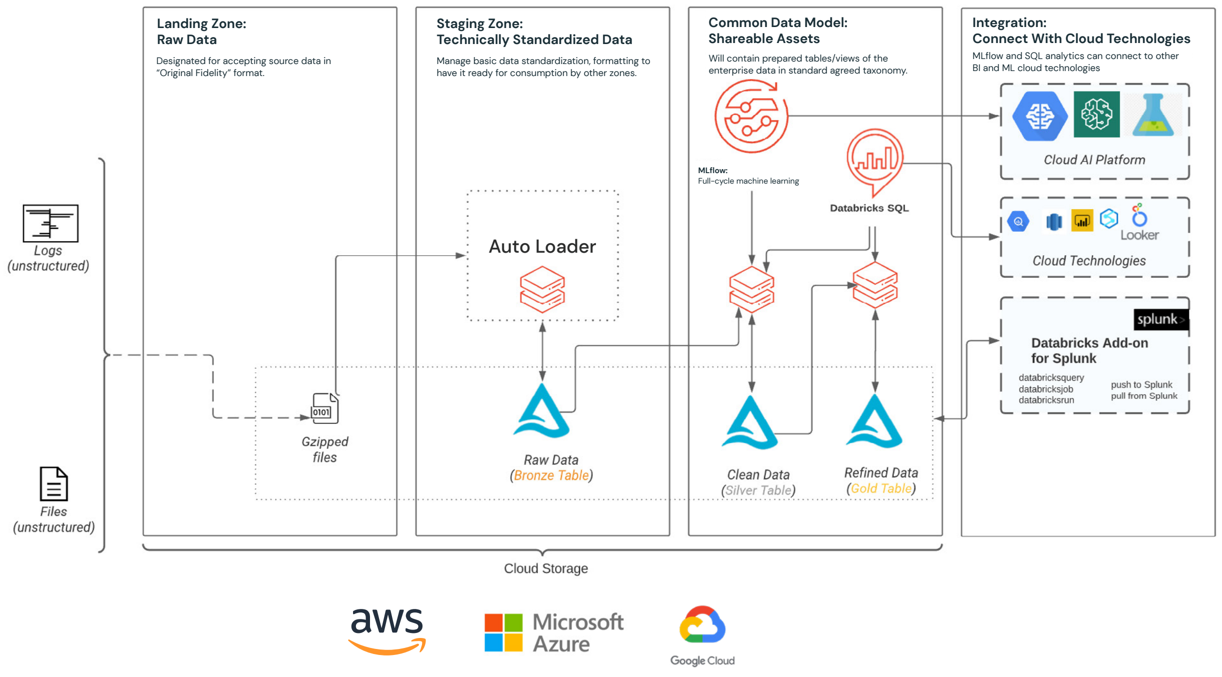

Landing Zone:
Raw Data
Designated for accepting source data in "Original Fidelity" format.
Staging Zone:
Technically Standardized Data
Manage basic data standardization, formatting to have it ready for consumption by other zones.
Common Data Model:
Shareable Assets
Will contain prepared tables/views of the enterprise data in standard agreed taxonomy.
Integration:
Connect With Cloud Technologies
MLflow and SQL analytics can connect to other BI and ML cloud technologies
Logs (unstructured)
Files (unstructured)
Gzipped files
Auto Loader
Raw Data (Bronze Table)
MLflow:
Full-cycle machine learning
Databricks SQL
Clean Data (Silver Table)
Refined Data (Gold Table)
Cloud AI Platform
Looker
Cloud Technologies
splunk>
Databricks Add-on for Splunk
databricksquery
databricksjob
databricksrun
push to Splunk
pull from Splunk
Cloud Storage
aws
Microsoft Azure
Google Cloud

Figure 1
Lakehouse architecture for CrowdStrike Falcon data

As we move from the Bronze to the Silver stage, schema will be added to provide structure to the data. Since we are reading from a single source of truth, we are able to process all of the different event types and enforce the correct schema as they are written to their respective tables. The ability to enforce schemas at the Silver layer provides a solid foundation for building ML and analytical workloads.

The Gold stage, which aggregates data for faster query and performance in dashboards and BI tools, is optional, depending on the use case and data volumes. Alerts can be set to trigger when unexpected trends are observed.

Another optional feature is the Databricks Add-on for Splunk, which allows security teams to take advantage of Databricks cost-effective model and the power of AI without having to leave the comforts of Splunk. Customers can run ad hoc queries against Databricks from within a Splunk dashboard or search bar with the add-on. Users can also launch notebooks or jobs in Databricks through a Splunk dashboard or in response to a Splunk search. The Databricks integration is bidirectional, letting customers summarize noisy data or run detections in Databricks that show up in Splunk Enterprise Security. Customers can even run Splunk searches from within a Databricks notebook to prevent the need to duplicate data.

The Splunk and Databricks integration allows customers to reduce costs, expand the data sources they analyze and provide the results of a more robust analytics engine, all without changing the tools used by their staff day-to-day.

Code walkthrough

Since Auto Loader abstracts the most complex part of file-based data ingestion, a raw-to-Bronze ingestion pipeline can be created within a few lines of code. Below is a Scala code example for a Delta ingestion pipeline. CrowdStrike Falcon event records have one common field name: “event_simpleName.”

```
val crowdstrikeStream = spark.readStream
  .format(“cloudFiles”)
  .option(“cloudFiles.format”, “text”)    // text file doesn’t need schema
  .option(“cloudFiles.region”, “us-west-2”)
  .option(“cloudFiles.useNotifications”, “true”)
  .load(rawDataSource)
  .withColumn(“load_timestamp”, current_timestamp())
  .withColumn(“load_date”, to_date($”load_timestamp”))
  .withColumn(“eventType”, from_json($”value”, “struct”, Map.empty[String, String]))
.selectExpr(“eventType.event_simpleName”,”load_date”,”load_timestamp”, “value” )
  .writeStream
  .format(“delta”)
  .option(“checkpointLocation”, checkPointLocation)
  .table(“demo_bronze.crowdstrike”)
```

In the raw-to-Bronze layer, only the event name is extracted from the raw data. By adding a load timestamp and date columns, users store the raw data into the Bronze table. The Bronze table is partitioned by event name and load date, which helps to make Bronze-to-Silver jobs more performant, especially when there is interest for a limited number of event date ranges. Next, a Bronze-to-Silver streaming job reads events from a Bronze table, enforces a schema and writes to hundreds of event tables based on the event name. Below is a Scala code example:

```
spark
  .readStream
  .option(“ignoreChanges”, “true”)
  .option(“maxBytesPerTrigger”, “2g”)
  .option(“maxFilesPerTrigger”, “64”)
  .format(“delta”)
  .load(bronzeTableLocation)
  .filter($”event_simpleName” === “event_name”)
  .withColumn(“event”, from_json($”value”, schema_of_json(sampleJson)) )
  .select($”event.*”, $”load_timestamp”, $”load_date”)
  .withColumn(“silver_timestamp”, current_timestamp())
  .writeStream
  .format(“delta”)
  .outputMode(“append”)
  .option(“mergeSchema”, “true”)
  .option(“checkpointLocation”, checkPoint)
  .option(“path”, tableLocation)
  .start()
```

Each event schema can be stored in a schema registry or in a Delta table in case a schema needs to be shared across multiple data-driven services. Note that the above code uses a sample JSON string read from the Bronze table, and the schema is inferred from the JSON using schema_of_json(). Later, the JSON string is converted to a struct using from_json(). Then, the struct is flattened, prompting the addition of a timestamp column. These steps provide a DataFrame with all the required columns to be appended to an event table. Finally, we write this structured data to an event table with append mode.

It is also possible to fan out events to multiple tables with one stream with foreachBatch by defining a function that will handle microbatches. Using foreachBatch(), it is possible to reuse existing batch data sources for filtering and writing to multiple tables. However, foreachBatch() provides only at-least-once write guarantees. So, a manual implementation is needed to enforce exactly-once semantics.

At this stage, the structured data can be queried with any of the languages supported in Databricks notebooks and jobs: Python, R, Scala and SQL. The Silver layer data is convenient to use for ML and cyberattack analysis.

The next streaming pipeline would be Silver-to-Gold. In this stage, it is possible to aggregate data for dashboarding and alerting. In the second part of this blog series we will provide some more insights into how we build dashboards using Databricks SQL.

What’s next

Stay tuned for more blog posts that build even more value on this use case by applying ML and using Databricks SQL.

You can use these notebooks in your own Databricks deployment. Each section of the notebooks has comments. We invite you to email us at cybersecurity@databricks.com. We look forward to your questions and suggestions for making this notebook easier to understand and deploy.

Start experimenting with these free Databricks notebooks.

SECTION 2.3

Unlocking the Power of Health Data With a Modern Data Lakehouse

by MICHAEL ORTEGA, MICHAEL SANKY and AMIR KERMANY

July 19, 2021

How to overcome the challenges of data warehouses and data lakes in the healthcare and life sciences industries

A single patient produces approximately 80 megabytes of medical data every year. Multiply that across thousands of patients over their lifetime, and you're looking at petabytes of patient data that contains valuable insights. Unlocking these insights can help streamline clinical operations, accelerate drug R&D and improve patient health outcomes. But first, the data needs to be prepared for downstream analytics and AI. Unfortunately, most healthcare and life sciences organizations spend an inordinate amount of time simply gathering, cleaning and structuring their data.

A single patient produces 80+ megabytes of medical data every year

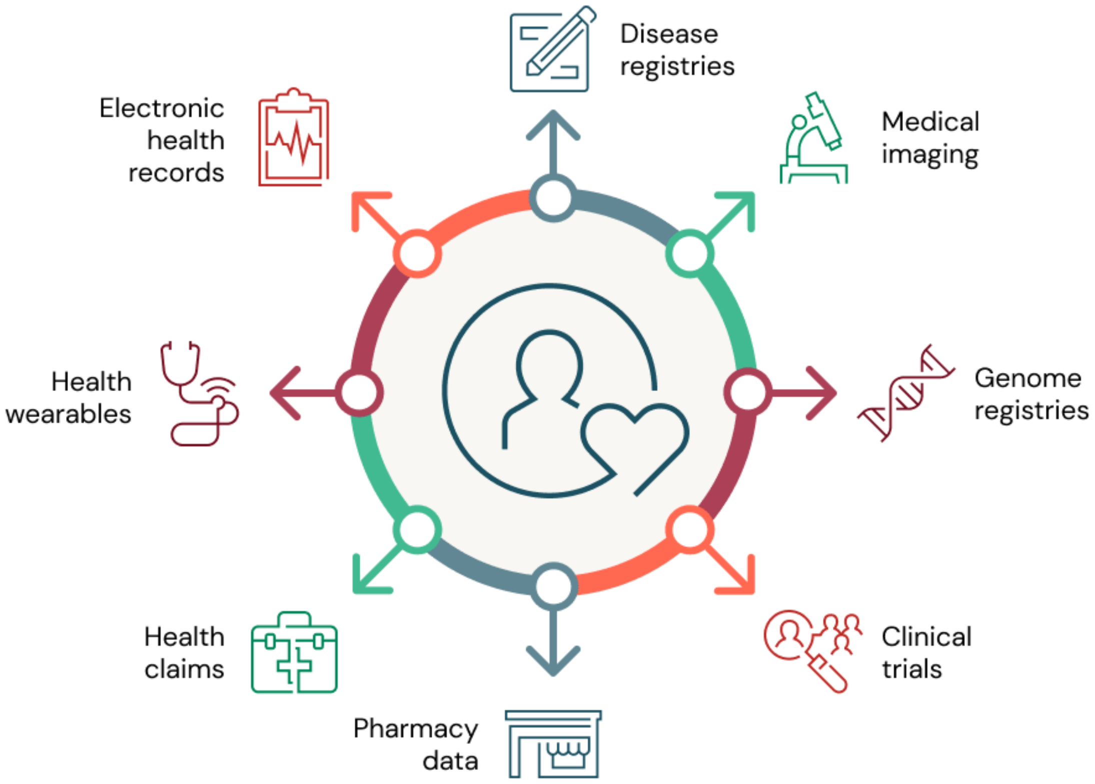


Figure 1
Health data is growing exponentially, with a single patient producing over 80 megabytes of data a year

Challenges with data analytics in healthcare and life sciences

There are lots of reasons why data preparation, analytics and AI are challenges for organizations in the healthcare industry, but many are related to investments in legacy data architectures built on data warehouses. Here are the four most common challenges we see in the industry:

CHALLENGE #1: VOLUME

Scaling for rapidly growing health data

Genomics is perhaps the single best example of the explosive growth in data volume in healthcare. The first genome cost more than $1B to sequence. Given the prohibitive costs, early efforts (and many efforts still) focused on genotyping, a process to look for specific variants in a very small fraction of a person's genome, typically around 0.1%. That evolved to Whole Exome Sequencing, which covers the protein coding portions of the genome, still less than 2% of the entire genome. Companies now offer direct-to-consumer tests for Whole Genome Sequencing (WGS) that are less than $300 for 30x WGS. On a population level, the UK Biobank is releasing more than 200,000 whole genomes for research this year. It's not just genomics. Imaging, health wearables and electronic medical records are growing tremendously as well.

Scale is the name of the game for initiatives like population health analytics and drug discovery. Unfortunately, many legacy architectures are built on-premises and designed for peak capacity. This approach results in unused compute power (and ultimately wasted dollars) during periods of low usage and doesn't scale quickly when upgrades are needed.

CHALLENGE #2: VARIETY

Analyzing diverse health data

Healthcare and life sciences organizations deal with a tremendous amount of data variety, each with its own nuances. It is widely accepted that over 80% of medical data is unstructured, yet most organizations still focus their attention on data warehouses designed for structured data and traditional SQL-based analytics. Unstructured data includes image data, which is critical to diagnose and measure disease progression in areas like oncology, immunology and neurology (the fastest growing areas of cost), and narrative text in clinical notes, which are critical to understanding the complete patient health and social history. Ignoring these data types, or setting them to the side, is not an option.

To further complicate matters, the healthcare ecosystem is becoming more interconnected, requiring stakeholders to grapple with new data types. For example, providers need claims data to manage and adjudicate risk-sharing agreements, and payers need clinical data to support processes like prior authorizations and to drive quality measures. These organizations often lack data architectures and platforms to support these new data types.

Some organizations have invested in data lakes to support unstructured data and advanced analytics, but this creates a new set of issues. In this environment, data teams now need to manage two systems — data warehouses and data lakes — where data is copied across siloed tools, resulting in data quality and management issues.

CHALLENGE #3: VELOCITY

Processing streaming data for real-time patient insights

In many settings, healthcare is a matter of life and death. Conditions can be very dynamic, and batch data processing — done even on a daily basis — often is not good enough. Access to the latest, up-to-the-second information is critical to successful interventional care. To save lives, streaming data is used by hospitals and national health systems for everything from predicting sepsis to implementing real-time demand forecasting for ICU beds.

Additionally, data velocity is a major component of the healthcare digital revolution. Individuals have access to more information than ever before and are able to influence their care in real time. For example, wearable devices — like the continuous glucose monitors provided by Livongo — stream real-time data into mobile apps that provide personalized behavioral recommendations.

Despite some of these early successes, most organizations have not designed their data architecture to accommodate streaming data velocity. Reliability issues and challenges integrating real-time data with historic data is inhibiting innovation.

CHALLENGE #:4 VERACITY

Building trust in healthcare data and AI

Last, but not least, clinical and regulatory standards demand the utmost level of data accuracy in healthcare. Healthcare organizations have high public health compliance requirements that must be met. Data democratization within organizations requires governance.

Additionally, organizations need good model governance when bringing artificial intelligence (AI) and machine learning (ML) into a clinical setting. Unfortunately, most organizations have separate platforms for data science workflows that are disconnected from their data warehouse. This creates serious challenges when trying to build trust and reproducibility in AI-powered applications.

Unlocking health data with a lakehouse

The lakehouse architecture helps healthcare and life sciences organizations overcome these challenges with a modern data architecture that combines the low cost, scalability and flexibility of a cloud data lake with the performance and governance of a data warehouse. With a lakehouse, organizations can store all types of data and power all types of analytics and ML in an open environment.

Building a Lakehouse for Healthcare and Life Sciences

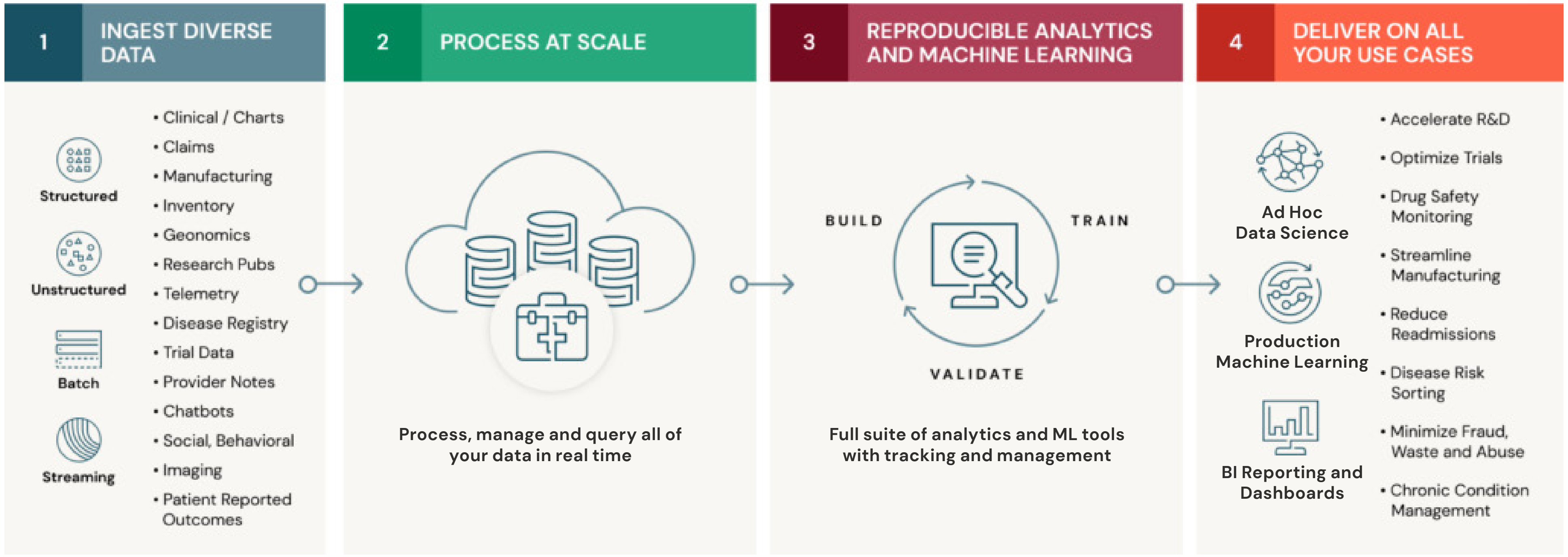


Figure 2
Deliver on all your healthcare and life sciences data analytics use cases with a modern lakehouse architecture

Specifically, the lakehouse provides the following benefits for healthcare and life sciences organizations:

- **Organize all your health data at scale.** At the core of the Databricks Lakehouse Platform is Delta Lake, an open-source data management layer that provides reliability and performance to your data lake. Unlike a traditional data warehouse, Delta Lake supports all types of structured and unstructured data, and to make ingesting health data easy, Databricks has built connectors for domain-specific data types like electronic medical records and genomics. These connectors come packaged with industry-standard data models in a set of quick-start Solution Accelerators. Additionally, Delta Lake provides built-in optimizations for data caching and indexing to significantly accelerate data processing speeds. With these capabilities, teams can land all their raw data in a single place and then curate it to create a holistic view of patient health.
- **Power all your patient analytics and AI.** With all your data centralized in a lakehouse, teams can build powerful patient analytics and predictive models directly on the data. To build on these capabilities, Databricks provides collaborative workspaces with a full suite of analytics and AI tools and support for a broad set of programming languages — such as SQL, R, Python and Scala. This empowers a diverse group of users, like data scientists, engineers and clinical informaticists, to work together to analyze, model and visualize all your health data.

- **Provide real-time patient insights.** The lakehouse provides a unified architecture for streaming and batch data. No need to support two different architectures nor wrestle with reliability issues. Additionally, by running the lakehouse architecture on Databricks, organizations have access to a cloud-native platform that auto-scales based on workload. This makes it easy to ingest streaming data and blend with petabytes of historic data for near real-time insights at population scale.
- **Deliver data quality and compliance.** To address data veracity, the lakehouse includes capabilities missing from traditional data lakes like schema enforcement, auditing, versioning and fine-grained access controls. An important benefit of the lakehouse is the ability to perform both analytics and ML on this same, trusted data source. Additionally, Databricks provides ML model tracking and management capabilities to make it easy for teams to reproduce results across environments and help meet compliance standards. All of these capabilities are provided in a HIPAA-compliant analytics environment.

This lakehouse is the best architecture for managing healthcare and life sciences data. By marrying this architecture with the capabilities of Databricks, organizations can support a wide range of highly impactful use cases, from drug discovery through chronic disease management programs.

Get started building your lakehouse for healthcare and life sciences

As mentioned above, we are pleased to make available a series of Solution Accelerators to help healthcare and life sciences organizations get started building a lakehouse for their specific needs. Our Solution Accelerators include sample data, prebuilt code and step-by-step instructions within a Databricks notebook.

New Solution Accelerator: Lakehouse for Real-World Evidence. Real-world data provides pharmaceutical companies with new insights into patient health and drug efficacy outside of a trial. This accelerator helps you build a Lakehouse for Real-World Evidence on Databricks. We'll show you how to ingest sample EHR data for a patient population, structure the data using the OMOP common data model and then run analyses at scale for challenges like investigating drug prescription patterns.

Start experimenting with these free Databricks notebooks.

Learn more about all of our Healthcare and Life Sciences solutions.

SECTION 2.4

Timeliness and Reliability in the Transmission of Regulatory Reports

by ANTOINE AMEND and FAHMID KABIR

September 17, 2021

Managing risk and regulatory compliance is an increasingly complex and costly endeavor. Regulatory change has increased 500% since the 2008 global financial crisis and boosted the regulatory costs in the process. Given the fines associated with non-compliance and SLA breaches (banks hit an all-time high in fines of $10 billion in 2019 for AML), processing reports has to proceed even if data is incomplete. On the other hand, a track record of poor data quality is also fined because of "insufficient controls." As a consequence, many financial services institutions (FSIs) are often left battling between poor data quality and strict SLAs, balancing between data reliability and data timeliness.

In this regulatory reporting Solution Accelerator, we demonstrate how Delta Live Tables can guarantee the acquisition and processing of regulatory data in real time to accommodate regulatory SLAs. With Delta Sharing and Delta Live Tables combined, analysts gain real-time confidence in the quality of regulatory data being transmitted. In this blog post, we demonstrate the benefits of the lakehouse architecture to combine financial services industry data models with the flexibility of cloud computing to enable high governance standards with low development overhead. We will now explain what a FIRE data model is and how Delta Live Tables can be integrated to build robust data pipelines.

FIRE data model

The Financial Regulatory data standard (FIRE) defines a common specification for the transmission of granular data between regulatory systems in finance. Regulatory data refers to data that underlies regulatory submissions, requirements and calculations and is used for policy, monitoring and supervision purposes. The FIRE data standard is supported by the European Commission, the Open Data Institute and the Open Data Incubator FIRE data standard for Europe via the Horizon 2020 funding program. As part of this solution, we contributed a PySpark module that can interpret FIRE data models into Apache Spark™ operating pipelines.

Delta Live Tables

Databricks recently announced a new product for data pipelines orchestration, Delta Live Tables, which makes it easy to build and manage reliable data pipelines at enterprise scale. With the ability to evaluate multiple expectations, discard or monitor invalid records in real time, the benefits of integrating the FIRE data model on Delta Live Tables are obvious. As illustrated in the following architecture, Delta Live Tables will **ingest** granular regulatory data landing onto cloud storage, **schematize** content and **validate** records for consistency in line with the FIRE data specification. Keep reading to see us demo the use of Delta Sharing to exchange granular information between regulatory systems in a safe, scalable and transparent manner.

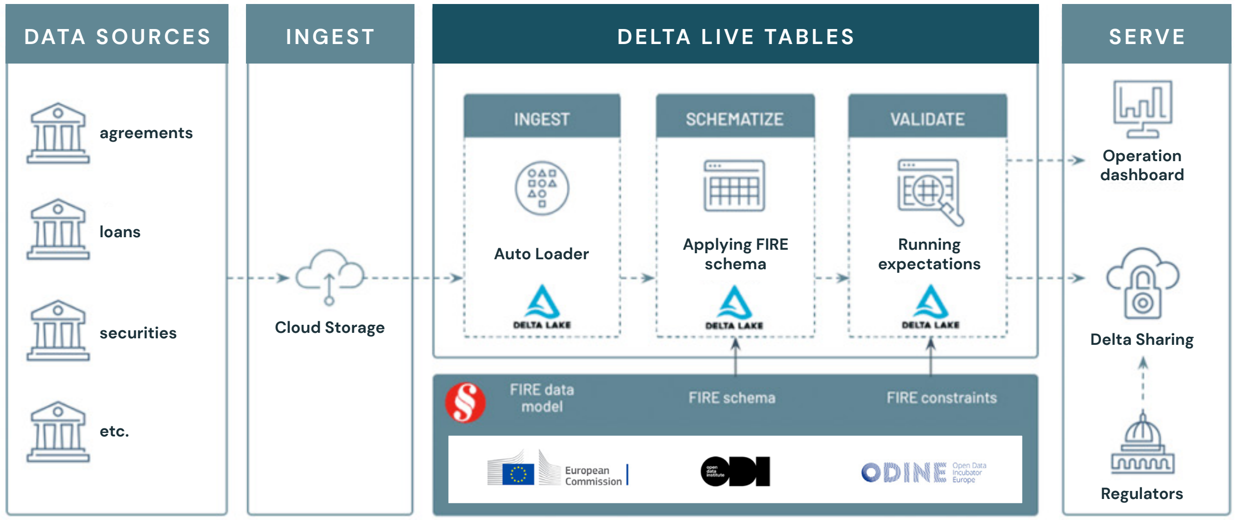


Figure 1

Enforcing schema

Even though some data formats may "look" structured (e.g., JSON files), enforcing a schema is not just a good engineering practice; in enterprise settings, and especially in the space of regulatory compliance, schema enforcement guarantees any missing field to be expected, unexpected fields to be discarded and data types to be fully evaluated (e.g., a date should be treated as a date object and not a string). It also proof-tests your systems for eventual data drift. Using the FIRE PySpark module, we programmatically retrieve the Spark schema required to process a given FIRE entity (collateral entity in that example) that we apply on a stream of raw records.

```
from fire.spark import FireModel
fire_model = FireModel().load("collateral")
fire_schema = fire_model.schema
```

In the example below, we enforce schema to incoming CSV files. By decorating this process using @dlt annotation, we define our entry point to our Delta Live Table, reading raw CSV files from a mounted directory and writing schematized records to a Bronze layer.

```
@dlt.create_table()
def collateral_bronze():
  return (
    spark
      .readStream
      .option("maxFilesPerTrigger", "1")
      .option("badRecordsPath", "/path/to/invalid/collateral")
      .format("csv")
      .schema(fire_schema)
      .load("/path/to/raw/collateral")
```

Evaluating expectations

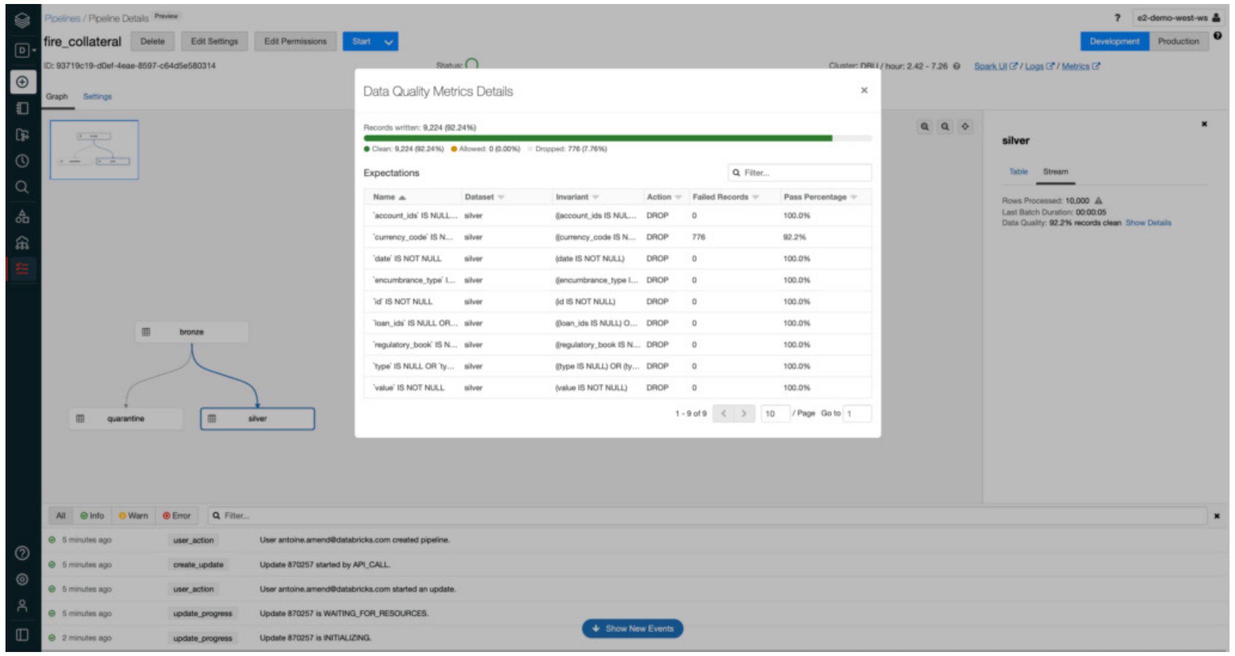
Applying a schema is one thing, enforcing its constraints is another. Given the schema definition of a FIRE entity (see example of the collateral schema definition), we can detect if a field is required or not. Given an enumeration object, we ensure its values are consistent (e.g., currency code). In addition to the technical constraints from the schema, the FIRE model also reports business expectations, such as minimum, maximum, monetary and maxItems. All these technical and business constraints will be programmatically retrieved from the FIRE data model and interpreted as a series of Spark SQL expressions.

```
from fire.spark import FireModel
fire_model = FireModel().load(“collateral”)
fire_constraints = fire_model.constraints
```

With Delta Live Tables, users have the ability to evaluate multiple expectations at once, enabling them to drop invalid records, simply monitor data quality or abort an entire pipeline. In our specific scenario, we want to drop records failing any of our expectations, which we later store to a quarantine table, as reported in the notebooks provided in this blog.

```
@dlt.create_table()
@dlt.expect_all_or_drop(fire_constraints)
def collateral_silver():
  return dlt.read_stream(“collateral_bronze”)
```

With only a few lines of code, we ensured that our Silver table is both syntactically (valid schema) and semantically (valid expectations) correct. As shown below, compliance officers have full visibility around the number of records being processed in real time. In this specific example, we ensured our collateral entity to be exactly 92.2% complete (quarantine handles the remaining 7.8%).



ID: 93719c19-d0ef-4eae-8597-c64d5e580314

Cluster: DBU / hour: 2.42 - 7.26

Spark UI / Logs / Metrics

Records written: 9,224 (92.24%)

● Clean: 9,224 (92.24%) ● Allowed: 0 (0.00%) ● Dropped: 776 (7.76%)

Name	Dataset	Invariant	Action	Failed Records	Pass Percentage
`account_ids` IS NULL...	silver	((account_ids IS NUL...	DROP	0	100.0%
`currency_code` IS N...	silver	((currency_code IS N...	DROP	776	92.2%
`date` IS NOT NULL	silver	(date IS NOT NULL)	DROP	0	100.0%
`encumbrance_type` I...	silver	((encumbrance_type I...	DROP	0	100.0%
`id` IS NOT NULL	silver	(id IS NOT NULL)	DROP	0	100.0%
`loan_ids` IS NULL OR...	silver	((loan_ids IS NULL) O...	DROP	0	100.0%
`regulatory_book` IS N...	silver	((regulatory_book IS N...	DROP	0	100.0%
`type` IS NULL OR `ty...	silver	((type IS NULL) OR (ty...	DROP	0	100.0%
`value` IS NOT NULL	silver	(value IS NOT NULL)	DROP	0	100.0%

1 - 9 of 9 10 / Page Go to 1

silver

Rows Processed: 10,000
Last Batch Duration: 00:00:05
Data Quality: 92.2% records clean

```
5 minutes ago    user_action        User antoine.amend@databricks.com created pipeline.
5 minutes ago    create_update      Update 870257 started by API_CALL.
5 minutes ago    user_action        User antoine.amend@databricks.com started an update.
5 minutes ago    update_progress    Update 870257 is WAITING_FOR_RESOURCES.
2 minutes ago    update_progress    Update 870257 is INITIALIZING.
```

Figure 2

Operations data store

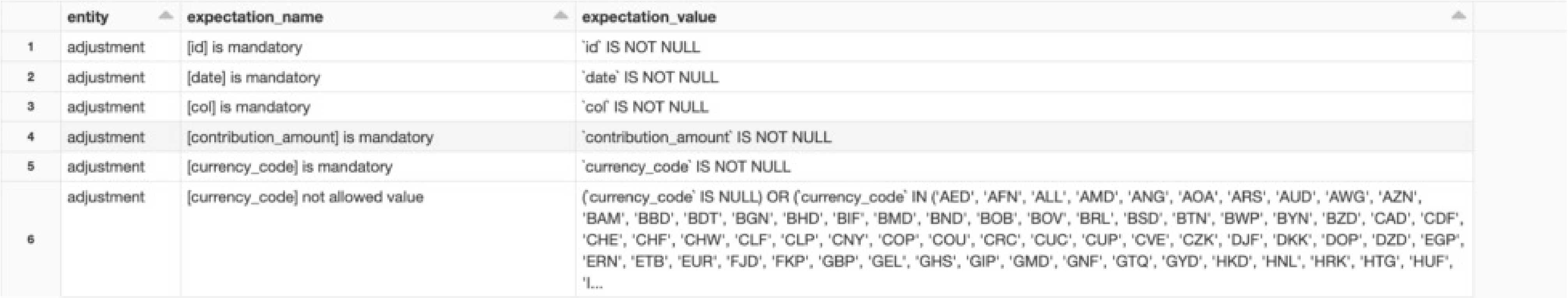
In addition to the actual data stored as Delta files, Delta Live Tables also stores operation metrics as “delta” format under system/events. We follow a standard pattern of the lakehouse architecture by “subscribing” to new operational metrics using Auto Loader, processing system events as new metrics unfold — in batch or in real time. Thanks to the transaction log of Delta Lake that keeps track of any data update, organizations can access new metrics without having to build and maintain their own checkpointing process.

```
input_stream = spark \
    .readStream \
    .format(“delta”) \
    .load(“/path/to/pipeline/system/events”)

output_stream = extract_metrics(input_stream)

output_stream \
    .writeStream \
    .format(“delta”) \
    .option(“checkpointLocation”, “/path/to/checkpoint”) \
    .table(metrics_table)
```

With all metrics available centrally into an operation store, analysts can use Databricks SQL to create simple dashboarding capabilities or more complex alerting mechanisms to detect data quality issues in real time.



	entity	expectation_name	expectation_value
1	adjustment	[id] is mandatory	`id` IS NOT NULL
2	adjustment	[date] is mandatory	`date` IS NOT NULL
3	adjustment	[col] is mandatory	`col` IS NOT NULL
4	adjustment	[contribution_amount] is mandatory	`contribution_amount` IS NOT NULL
5	adjustment	[currency_code] is mandatory	`currency_code` IS NOT NULL
6	adjustment	[currency_code] not allowed value	(`currency_code` IS NULL) OR (`currency_code` IN ('AED', 'AFN', 'ALL', 'AMD', 'ANG', 'AOA', 'ARS', 'AUD', 'AWG', 'AZN', 'BAM', 'BBD', 'BDT', 'BGN', 'BHD', 'BIF', 'BMD', 'BND', 'BOB', 'BOV', 'BRL', 'BSD', 'BTN', 'BWP', 'BYN', 'BZD', 'CAD', 'CDF', 'CHE', 'CHF', 'CHW', 'CLF', 'CLP', 'CNY', 'COP', 'COU', 'CRC', 'CUC', 'CUP', 'CVE', 'CZK', 'DJF', 'DKK', 'DOP', 'DZD', 'EGP', 'ERN', 'ETB', 'EUR', 'FJD', 'FKP', 'GBP', 'GEL', 'GHS', 'GIP', 'GMD', 'GNF', 'GTQ', 'GYD', 'HKD', 'HNL', 'HRK', 'HTG', 'HUF', 'I...

Figure 3

The immutability aspect of the Delta Lake format coupled with the transparency in data quality offered by Delta Live Tables allows financial institutions to “time travel” to specific versions of their data that matches both volume and quality required for regulatory compliance. In our specific example, replaying our 7.8% of invalid records stored in quarantine will result in a different Delta version attached to our Silver table, a version that can be shared amongst regulatory bodies.

```
DESCRIBE HISTORY fire.collateral_silver
```

Transmission of regulatory data

With full confidence in both data quality and volume, financial institutions can safely exchange information between regulatory systems using Delta Sharing, an open protocol for enterprise data exchange. Not constraining end users to the same platform nor relying on complex ETL pipelines to consume data (accessing data files through an SFTP server, for instance), the open source nature of Delta Lake makes it possible for data consumers to access schematized data natively from Python, Spark or directly through MI/BI dashboards (such as Tableau or Power BI).

Although we could be sharing our Silver table as is, we may want to use business rules that only share regulatory data when a predefined data quality threshold is met. In this example, we clone our Silver table at a different version and to a specific location segregated from our internal networks and accessible by end users (demilitarized zone, or DMZ).

```
from delta.tables import *

deltaTable = DeltaTable.forName(spark, “fire.collateral_silver”)
deltaTable.cloneAtVersion(
  approved_version,
  dmz_path,
  isShallow=False,
  replace=True
)

spark.sql(
  “CREATE TABLE fire.colleral_gold USING DELTA LOCATION ‘{}’”
    .format(dmz_path)
)
```

Although the Delta Sharing open source solution relies on a sharing server to manage permission, Databricks leverages Unity Catalog to centralize and enforce access control policies, provide users with full audit logs capability and simplify access management through its SQL interface. In the example below, we create a SHARE that includes our regulatory tables and a RECIPIENT to share our data with.

```
-- DEFINE OUR SHARING STRATEGY
CREATE SHARE regulatory_reports;

ALTER SHARE regulatory_reports ADD TABLE fire.collateral_gold;
ALTER SHARE regulatory_reports ADD TABLE fire.loan_gold;
ALTER SHARE regulatory_reports ADD TABLE fire.security_gold;
ALTER SHARE regulatory_reports ADD TABLE fire.derivative_gold;

-- CREATE RECIPIENTS AND GRANT SELECT ACCESS
CREATE RECIPIENT regulatory_body;

GRANT SELECT ON SHARE regulatory_reports TO RECIPIENT regulatory_body;
```

Any regulator or user with granted permissions can access our underlying data using a personal access token exchanged through that process. For more information about Delta Sharing, please visit our product page and contact your Databricks representative.

Proof-test your compliance

Through this series of notebooks and Delta Live Tables jobs, we demonstrated the benefits of the lakehouse architecture in the ingestion, processing, validation and transmission of regulatory data. Specifically, we addressed the need for organizations to ensure consistency, integrity and timeliness of regulatory pipelines that could be easily achieved using a common data model (FIRE) coupled with a flexible orchestration engine (Delta Live Tables). With Delta Sharing capabilities, we finally demonstrated how FSIs could bring full transparency and confidence to the regulatory data exchanged between various regulatory systems while meeting reporting requirements, reducing operation costs and adapting to new standards.

Get familiar with the FIRE data pipeline using the attached notebooks and visit our Solution Accelerators Hub to get up to date with our latest solutions for financial services.

Start experimenting with these free Databricks notebooks.

SECTION 2.5

AML Solutions at Scale Using Databricks Lakehouse Platform

by SRI GHATTAMANENI, RICARDO PORTILLA and ANINDITA MAHAPATRA

July 16, 2021

Solving the key challenges to building a financial crimes solution

Anti-money laundering (AML) compliance has been undoubtedly one of the top agenda items for regulators providing oversight of financial institutions across the globe. As AML evolved and became more sophisticated over the decades, so have the regulatory requirements designed to counter modern money laundering and terrorist financing schemes. The Bank Secrecy Act of 1970 provided guidance and framework for financial institutions to put in proper controls to monitor financial transactions and report suspicious fiscal activity to relevant authorities. This law provided the framework for how financial institutes combat money laundering and financial terrorism.

Why anti-money laundering is so complex

Current AML operations bear little resemblance to those of the last decade. The shift to digital banking, with financial institutions (FIs) processing billions of transactions daily, has resulted in the ever increasing scope of money laundering, even with stricter transaction monitoring systems and robust Know Your Customer (KYC) solutions. In this blog, we share our experiences working with our FI customers to build enterprise-scale AML solutions on the lakehouse platform that both provides strong oversight and delivers innovative, scalable solutions to adapt to the reality of modern online money laundering threats.

Building an AML solution with lakehouse

The operational burden of processing billions of transactions a day comes from the need to store the data from multiple sources and power intensive, next-gen AML solutions. These solutions provide powerful risk analytics and reporting while supporting the use of advanced machine learning models to reduce false positives and improve downstream investigation efficiency. FIs have already taken steps to solve the infrastructure and scaling problems by moving from on-premises to cloud for better security, agility and the economies of scale required to store massive amounts of data.

But then there is the issue of how to make sense of the massive amounts of structured and unstructured data collected and stored on cheap object storage. While cloud vendors provide an inexpensive way to store the data, making sense of the data for downstream AML risk management and compliance activities starts with storage of the data in high-quality and performant formats for downstream consumption. The Databricks Lakehouse Platform does exactly this. By combining the low storage cost benefits of data lakes with the robust transaction capabilities of data warehouses, FIs can truly build the modern AML platform.

On top of the data storage challenges outlined above, AML analysts face some key domain-specific challenges:

- Improve time-to-value parsing unstructured data such as images, textual data and network links
- Reduce DevOps burden for supporting critical ML capabilities such as entity resolution, computer vision and graph analytics on entity metadata
- Break down silos by introducing analytics engineering and a dashboarding layer on AML transactions and enriched tables

Luckily, Databricks helps solve these by leveraging Delta Lake to store and combine both unstructured and structured data to build entity relationships; moreover, Databricks Delta Engine provides efficient access using the new Photon compute to speed up BI queries on tables. On top of these capabilities, ML is a first-class citizen in lakehouse, which means analysts and data scientists do not waste time subsampling or moving data to share dashboards and stay one step ahead of bad actors.

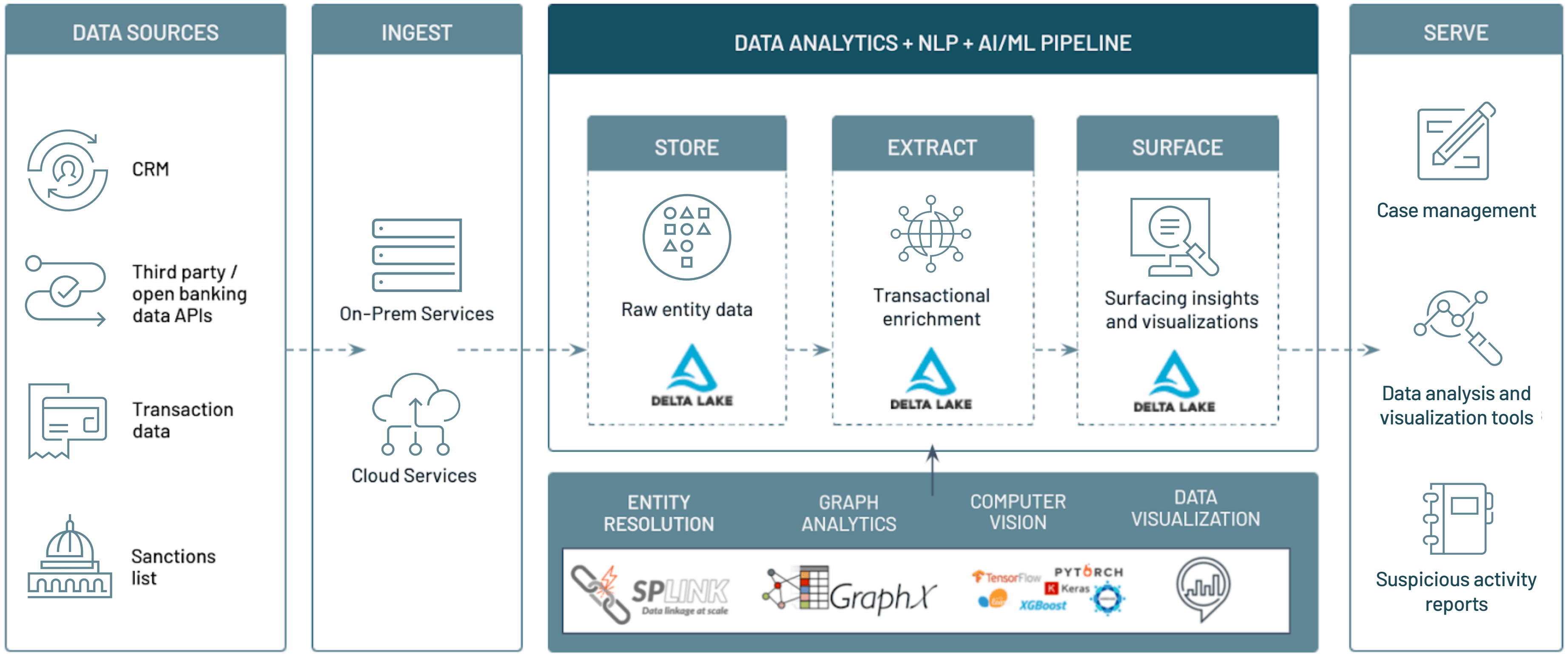


Figure 1

Detecting AML patterns with graph capabilities

One of the main data sources that AML analysts use as part of a case is *transaction data*. Even though this data is tabular and easily accessible with SQL, it becomes cumbersome to track chains of transactions that are three or more layers deep with SQL queries. For this reason, it is important to have a flexible suite of languages and APIs to express simple concepts such as a connected network of suspicious individuals transacting illegally together. Luckily, this is simple to accomplish using GraphFrames, a graph API pre-installed in the Databricks Runtime for Machine Learning.

In this section, we will show how graph analytics can be used to detect AML schemes such as synthetic identity and layering / structuring. We are going to utilize a data set consisting of transactions, as well as entities derived from transactions, to detect the presence of these patterns with Apache Spark™, GraphFrames and Delta Lake. The persisted patterns are saved in Delta Lake so that Databricks SQL can be applied on the Gold-level aggregated versions of these findings, offering the power of graph analytics to end users.

Scenario 1 — synthetic identities

As mentioned above, the existence of synthetic identities can be a cause for alarm. Using graph analysis, all of the entities from our transactions can be analyzed in bulk to detect a risk level. In our analysis, this is done in three phases:

- Based on the transaction data, extract the entities
- Create links between entities based on address, phone number or email
- Use GraphFrames-connected components to determine whether multiple entities (identified by an ID and other attributes above) are connected via one or more links

Based on how many connections (i.e., common attributes) exist between entities, we can assign a lower or higher risk score and create an alert based on high-scoring groups. Below is a basic representation of this idea.

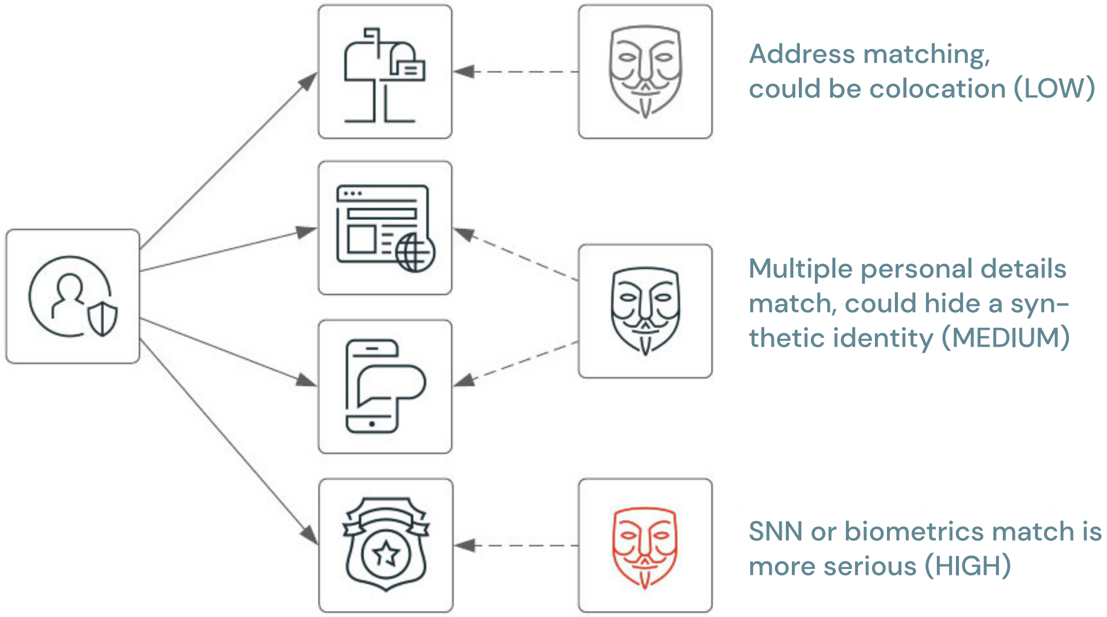

Address matching, could be colocation (LOW)
Multiple personal details match, could hide a synthetic identity (MEDIUM)
SNN or biometrics match is more serious (HIGH)

Figure 2

First, we create an identity graph using an address, email and phone number to link individuals if they match any of these attributes.

```
e_identity_sql = '''
select entity_id as src, address as dst from aml.aml_entities_synth  where address is not
null
UNION
select entity_id as src, email as dst from aml.aml_entities_synth  where email_addr is not
null
UNION
select entity_id as src, phone as dst from aml.aml_entities_synth  where phone_number is not
null
'''

from graphframes import *
from pyspark.sql.functions import *
aml_identity_g = GraphFrame(identity_vertices, identity_edges)
result = aml_identity_g.connectedComponents()

result \
 .select("id", "component", 'type') \
 .createOrReplaceTempView("components")
```

Next, we'll run queries to identify when two entities have overlapping personal identification and scores. Based on the results of these querying graph components, we would expect a cohort consisting of only one matching attribute (such as address), which isn't too much cause for concern. However, as more attributes match, we should expect to be alerted. As shown below, we can flag cases where all three attributes match, allowing SQL analysts to get daily results from graph analytics run across all entities.

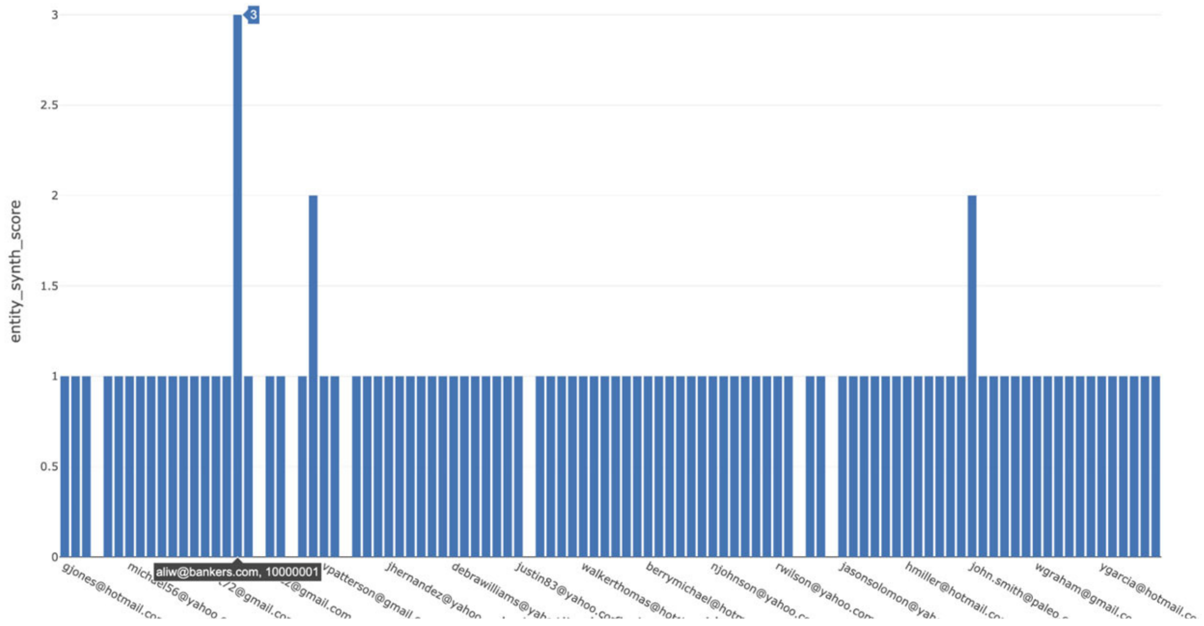


Figure 3

Scenario 2 — structuring

Another common pattern is called *structuring*, which occurs when multiple entities collude and send smaller “under the radar” payments to a set of banks, which subsequently route larger aggregate amounts to a final institution (as depicted below on the far right). In this scenario, all parties have stayed under the $10,000 threshold amount, which would typically alert authorities. Not only is this easily accomplished with graph analytics, but the *motif finding technique* can be automated to extend to other permutations of networks and locate other suspicious transactions in the same way.

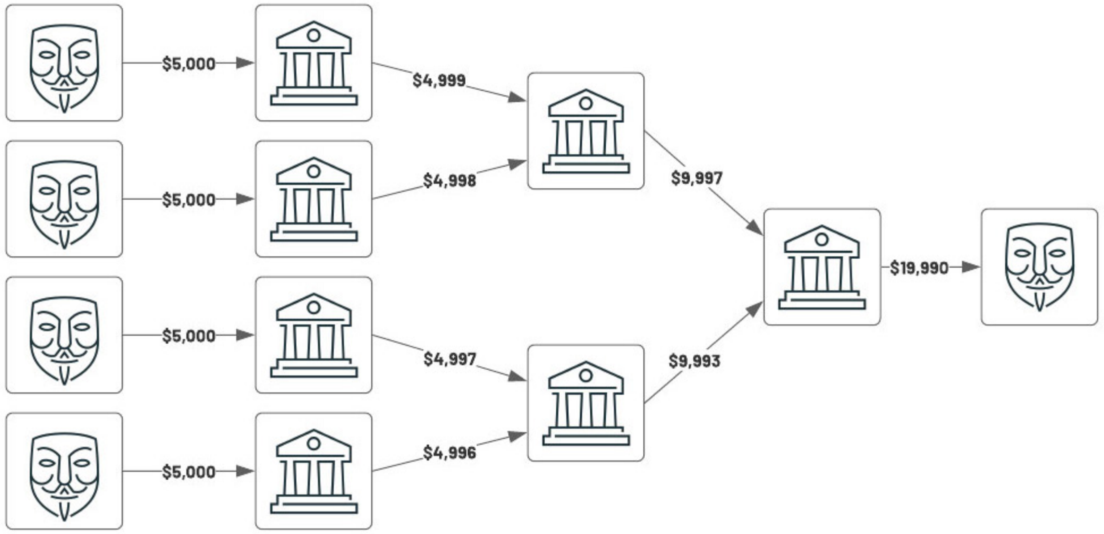


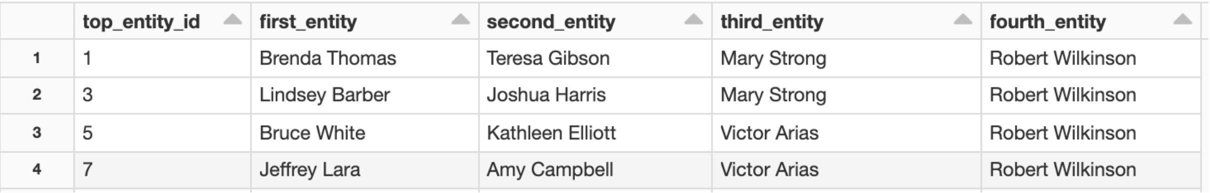
Figure 4

Now we’ll write the basic motif-finding code to detect the scenario above using graph capabilities. Note that the output here is semi-structured JSON; all data types, including unstructured types, are easily accessible in the lakehouse — we will save these particular results for SQL reporting.

```
motif = “(a)-[e1]->(b); (b)-[e2]->(c); (c)-[e3]->(d); (e)-[e4]->(f); (f)-[e5]-
>(c); (c)-[e6]->(g)”
struct_scn_1 = aml_entity_g.find(motif)

joined_graphs = struct_scn_1.alias(“a”) \
 .join(struct_scn_1.alias(“b”), col(“a.g.id”) == col(“b.g.id”)) \
 .filter(col(“a.e6.txn_amount”) + col(“b.e6.txn_amount”) > 10000)
```

Using motif finding, we extracted interesting patterns where money is flowing through four different entities and kept under a $10,000 threshold. We join our graph metadata back to structured data sets to generate insights for an AML analyst to investigate further.

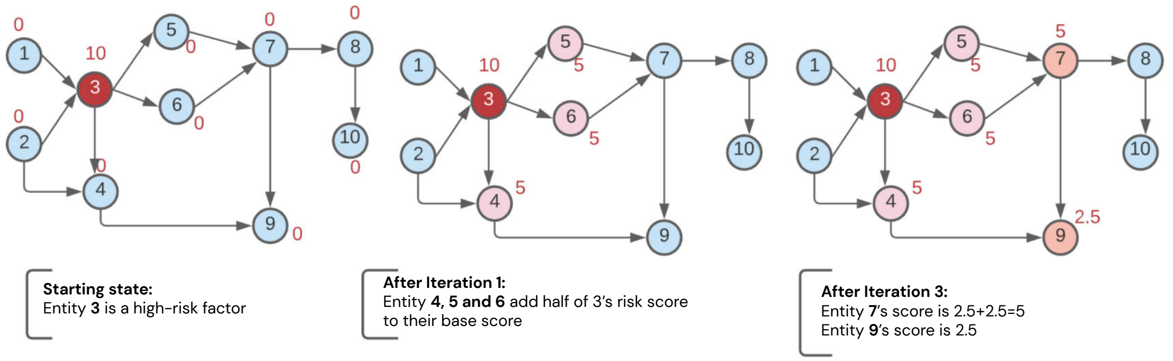


	top_entity_id	first_entity	second_entity	third_entity	fourth_entity
1	1	Brenda Thomas	Teresa Gibson	Mary Strong	Robert Wilkinson
2	3	Lindsey Barber	Joshua Harris	Mary Strong	Robert Wilkinson
3	5	Bruce White	Kathleen Elliott	Victor Arias	Robert Wilkinson
4	7	Jeffrey Lara	Amy Campbell	Victor Arias	Robert Wilkinson

Figure 5

Scenario 3 — risk score propagation

The identified high-risk entities will have an influence (a network effect) on their circle. So, the risk score of all the entities that they interact with must be adjusted to reflect the zone of influence. Using an iterative approach, we can follow the flow of transactions to any given depth and adjust the risk scores of others affected in the network. As mentioned previously, running graph analytics avoids multiple repeated SQL joins and complex business logic, which can impact performance due to memory constraints. Graph analytics and Pregel API was built for that exact purpose. Initially developed by Google, Pregel allows users to recursively "propagate" messages from any vertex to its corresponding neighbors, updating vertex state (their risk score here) at each step. We can represent our dynamic risk approach using Pregel API as follows.



Starting state:
Entity **3** is a high-risk factor

After Iteration 1:
Entity **4, 5 and 6** add half of 3's risk score to their base score

After Iteration 3:
Entity **7**'s score is 2.5+2.5=5
Entity **9**'s score is 2.5

Figure 6

The diagram below left shows the starting state of the network and two subsequent iterations. Say we started with one bad actor (Node 3) with a risk score of 10. We want to penalize all the people who transact with that node (namely Nodes 4, 5 and 6) and receive funds by passing on, for instance, half the risk score of the bad actor, which then is added to their base score. In the next iteration, all nodes that are downstream from Nodes 4, 5 and 6 will get their scores adjusted.

Node #	Iteration #0	Iteration #1	Iteration #2
1	0	0	0
2	0	0	0
3	10	10	10
4	0	5	5
5	0	5	5
6	0	5	5
7	0	0	5
8	0	0	0
9	0	0	2.5
10	0	0	0

Using the Pregel API from GraphFrame, we can do this computation and persist the modified scores for other applications downstream to consume.

```
from graphframes.lib import Pregel

ranks = aml_entity_g.pregel \
    .setMaxIter(3) \
    .withVertexColumn(
       “risk_score”,
       col(“risk”),
       coalesce(Pregel.msg()+ col(“risk”),
       col(“risk_score”))
    ) \
    .sendMsgToDst(Pregel.src(“risk_score”)/2 )  \
    .aggMsgs(sum(Pregel.msg())) \
    .run()
```

Address matching

A pattern we want to briefly touch upon is address matching of text to actual street view images. Oftentimes, there is a need for an AML analyst to validate the legitimacy of addresses that are linked to entities on file. Is this address a commercial building, a residential area or a simple postbox? However, analyzing pictures is often a tedious, time-consuming and manual process to obtain, clean and validate. A lakehouse data architecture allows us to automate most of this task using Python and ML runtimes with PyTorch and pretrained open source models. Below is an example of a valid address to the human eye. To automate validation, we will use a pretrained VGG model for which there are hundreds of valid objects we can use to detect a residence.

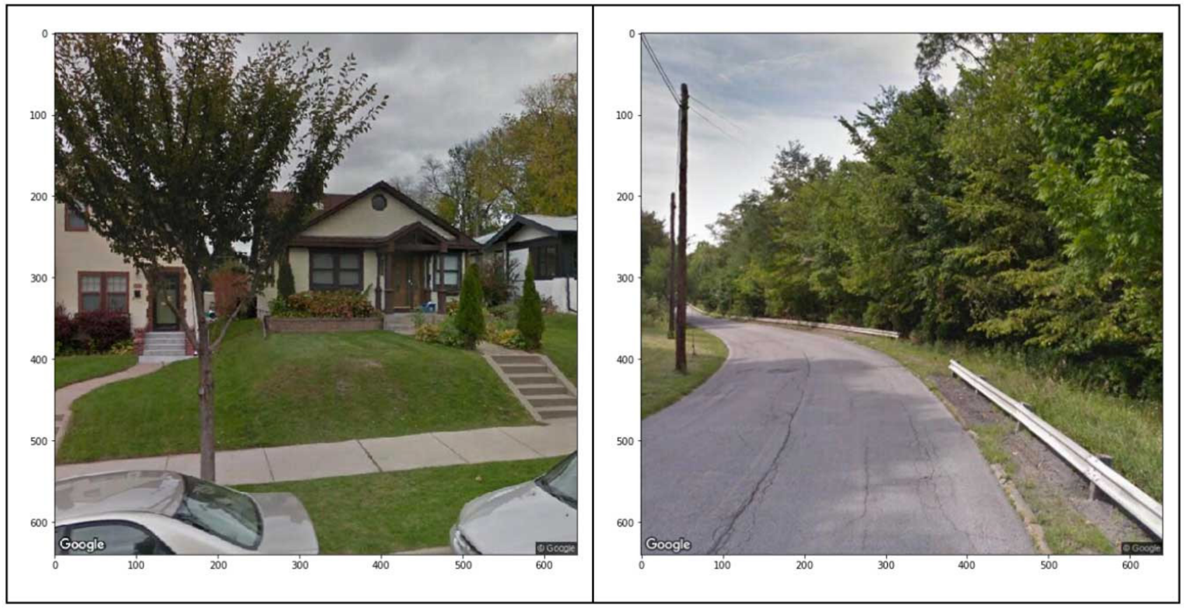


Figure 7

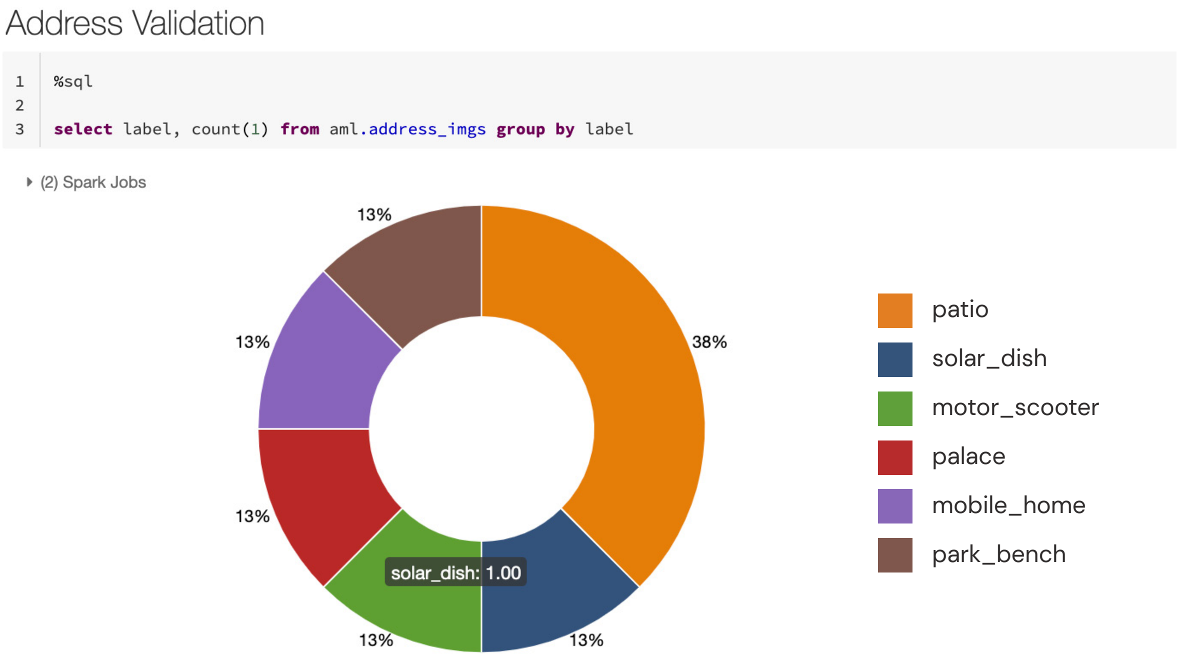
Using the code below, which can be automated to run daily, we'll now have a label attached to all our images — we've loaded all the image references and labels up into a SQL table for simpler querying also. Notice in the code below how simple it is to query a set of images for the objects inside them — the ability to query such unstructured data with Delta Lake is an enormous time-saver for analysts, and speeds up the validation process to minutes instead of days or weeks.

```
from PIL import Image
from matplotlib import cm

img = Image.fromarray(img)
...

vgg = models.vgg16(pretrained=True)
prediction = vgg(img)
prediction = prediction.data.numpy().argmax()
img_and_labels[i] = labels[prediction]
```

As we start to summarize, we notice some interesting categories appear. As seen in the breakdown below, there are a few obvious labels such as *patio*, *mobile home* and *motor scooter* we would expect to see as items detected in a residential address. On the other hand, the CV model has labeled a solar dish from surrounding objects in one image. (*Note: since we are restricted to an open source model not trained on a custom set of images, the solar dish label is not accurate*.) Upon further analysis of the image, we drill down and immediately see that i) there is not a real solar dish here and more importantly ii) this address is not a real residence (pictured in our side-by-side comparison on Figure 7). The Delta Lake format allows us to store a reference to our unstructured data along with a label for simple querying in our classification breakdown below.



Address Validation

```
%sql

select label, count(1) from aml.address_imgs group by label
```

Figure 8

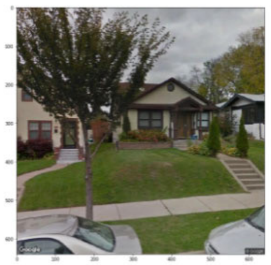
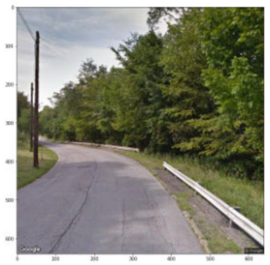
Image Name	Rendered Image	Main Object	Risk Level
img_0.jpg		Patio	Low
img_1.jpg		Solar Dish	High

Figure 9

Entity resolution

The last category of AML challenges that we'll focus on is entity resolution. Many open source libraries tackle this problem, so for some basic entity fuzzy matching, we chose to highlight Splink, which achieves the linkage at scale and offers configurations to specify matching columns and blocking rules.

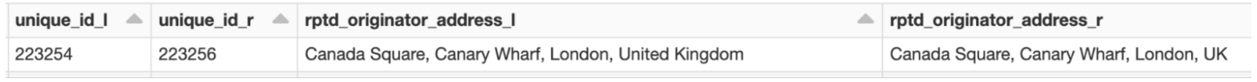
In the context of the entities derived from our transactions, it is a simple exercise to insert our Delta Lake transactions into the context of Splink.

```
settings = {
  “link_type”: “dedupe_only”,
  “blocking_rules”: [
      “l.txn_amount = r.txn_amount”,
  ],
  “comparison_columns”: [
      {
          “col_name”: “rptd_originator_address”,
      },
      {
          “col_name”: “rptd_originator_name”,
      }
  ]
}

from splink import Splink
linker = Splink(settings, df2, spark)
df2_e = linker.get_scored_comparisons()
```

Splink works by assigning a match probability that can be used to identify transactions in which entity attributes are highly similar, raising a potential alert with respect to a reported address, entity name or transaction amount. Given the fact that entity resolution can be highly manual for matching account information, having open source libraries that automate this task and save the information in Delta Lake can make investigators much more productive for case resolution. While there are several options available for entity matching, we recommend using Locality-Sensitive Hashing (LSH) to identify the right algorithm for the job. You can learn more about LSH and its benefits in this blog post.

As reported above, we quickly found some inconsistencies for the NY Mellon bank address, with “Canada Square, Canary Wharf, London, United Kingdom” similar to “Canada Square, Canary Wharf, London, UK.” We can store our de-duplicated records back to a Delta table that can be used for AML investigation.



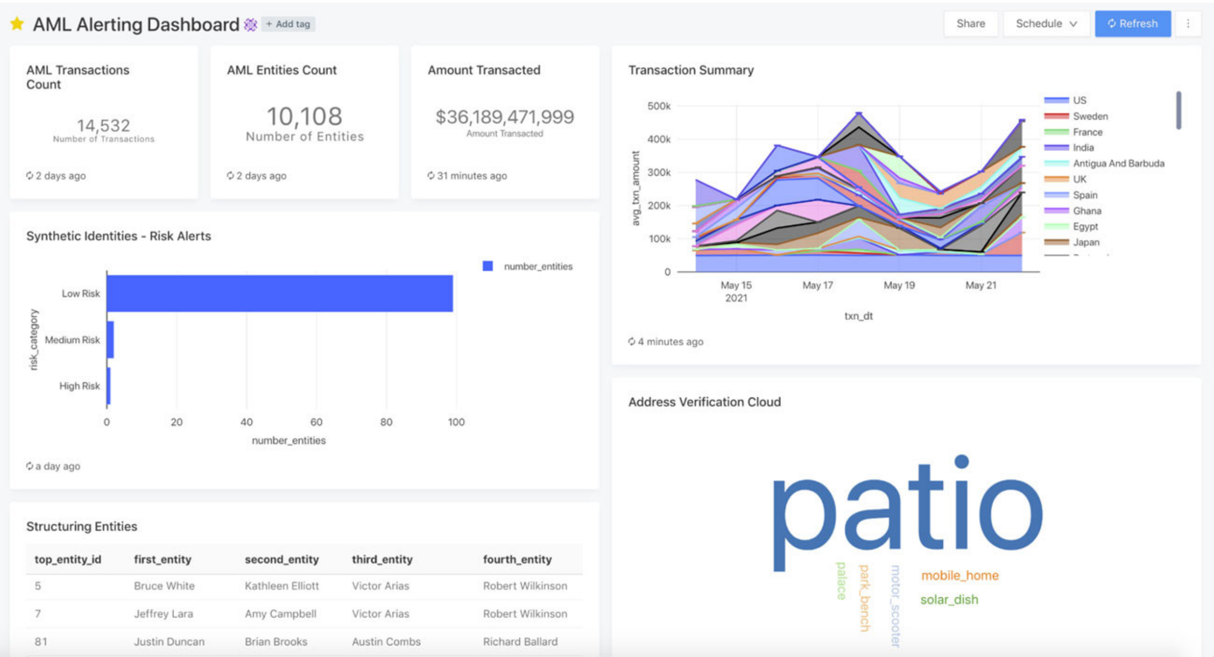
unique_id_l	unique_id_r	rptd_originator_address_l	rptd_originator_address_r
223254	223256	Canada Square, Canary Wharf, London, United Kingdom	Canada Square, Canary Wharf, London, UK

Figure 10

AML lakehouse dashboard

Databricks SQL on the lakehouse is closing the gap with respect to traditional data warehouses in terms of simplified data management, performance with new query engine Photon and user concurrency. This is important since many organizations do not have the budget for overpriced proprietary AML software to support the myriad use cases, such as combatting the financing of terrorism (CFT), that help fight financial crime. In the market, there are dedicated solutions that can perform the graph analytics above, dedicated solutions to address BI in a warehouse and dedicated solutions for ML. The AML lakehouse design unifies all three. AML data platform teams can leverage Delta Lake at the lower cost of cloud storage while easily integrating open source technologies to produce curated reports based on graph technology, computer vision and SQL analytics engineering. In figure 11, we will show a materialization of the reporting for AML.

The attached notebooks produced a transactions object, entities object, as well as summaries such as structuring prospects, synthetic identity tiers and address classifications using pretrained models. In the Databricks SQL visualization below, we used our Photon SQL engine to execute summaries on these and built-in visualization to produce a reporting dashboard within minutes. There are full ACLs on both tables, as well as the dashboard itself, to allow users to share with executives and data teams — a scheduler to run this report periodically is also built-in. The dashboard is a culmination of AI, BI and analytics engineering built into the AML solution.



AML Alerting Dashboard

AML Transactions Count
14,532
Number of Transactions
2 days ago

AML Entities Count
10,108
Number of Entities
2 days ago

Amount Transacted
$36,189,471,999
Amount Transacted
31 minutes ago

Synthetic Identities - Risk Alerts
a day ago

Transaction Summary
4 minutes ago

Address Verification Cloud

Structuring Entities

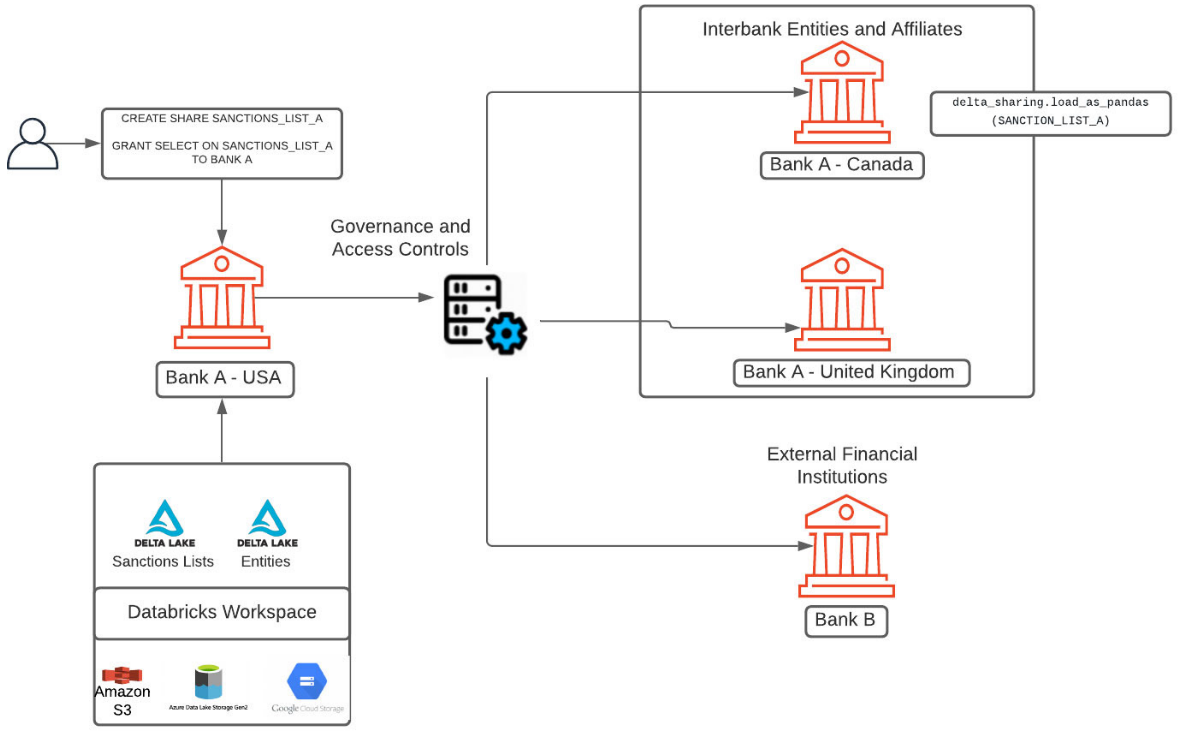
top_entity_id	first_entity	second_entity	third_entity	fourth_entity
5	Bruce White	Kathleen Elliott	Victor Arias	Robert Wilkinson
7	Jeffrey Lara	Amy Campbell	Victor Arias	Robert Wilkinson
81	Justin Duncan	Brian Brooks	Austin Combs	Richard Ballard

Figure 11

The open banking transformation

The rise of open banking enables FIs to provide a better customer experience via data sharing between consumers, FIs and third-party service providers through APIs. An example of this is Payment Services Directive (PSD2), which transformed financial services in the EU region as part of Open Banking Europe regulation. As a result, FIs have access to more data from multiple banks and service providers, including customer account and transaction data. This trend has expanded within the world of fraud and financial crimes with the latest guidance from FinCEN under section 314(b) of USA Patriot Act; covered FIs can now share information with other FIs and within domestic and foreign branches regarding individuals, entities, organizations and so on that are suspected to be involved in potential money laundering.

While information sharing provision helps with transparency and protects the United States financial systems against money laundering and terrorism financing, the information exchange must be done using protocols with proper data and security protections. To solve the problem of securing information sharing, Databricks recently announced Delta Sharing, an open and secure protocol for data sharing. Using familiar open source APIs, such as pandas and Spark, data producers and consumers can now share data using secure and open protocols and maintain a full audit of all the data transactions to maintain compliance with FinCEN regulations.



```
CREATE SHARE SANCTIONS_LIST_A

GRANT SELECT ON SANCTIONS_LIST_A
TO BANK A
```

```
delta_sharing.load_as_pandas
(SANCTION_LIST_A)
```

Figure 12

Conclusion

The lakehouse architecture is the most scalable and versatile platform to enable analysts in their AML analytics. Lakehouse supports use cases ranging from fuzzy match to image analytics to BI with built-in dashboards, and all of these capabilities will allow organizations to reduce total cost of ownership compared to proprietary AML solutions. The financial services team at Databricks is working on a variety of business problems in the financial services space and enabling data engineering and data science professionals to start the Databricks journey through Solution Accelerators like AML.

Start experimenting with these free Databricks notebooks

- Introduction to graph theory for AML
- Introduction to computer vision for AML
- Introduction to entity resolution for AML

SECTION 2.6

Build a Real-Time AI Model to Detect Toxic Behavior in Gaming

by DAN MORRIS and DUNCAN DAVIS

June 16, 2021

Across massively multiplayer online video games (MMOs), multiplayer online battle arena games (MOBAs) and other forms of online gaming, players continuously interact in real time to either coordinate or compete as they move toward a common goal — winning. This interactivity is integral to game play dynamics, but at the same time, it's a prime opening for toxic behavior — an issue pervasive throughout the online video gaming sphere.

Toxic behavior manifests in many forms, such as the varying degrees of griefing, cyberbullying and sexual harassment that are illustrated in the matrix above right from Behaviour Interactive, which lists the types of interactions seen within the multiplayer game "Dead by Night."

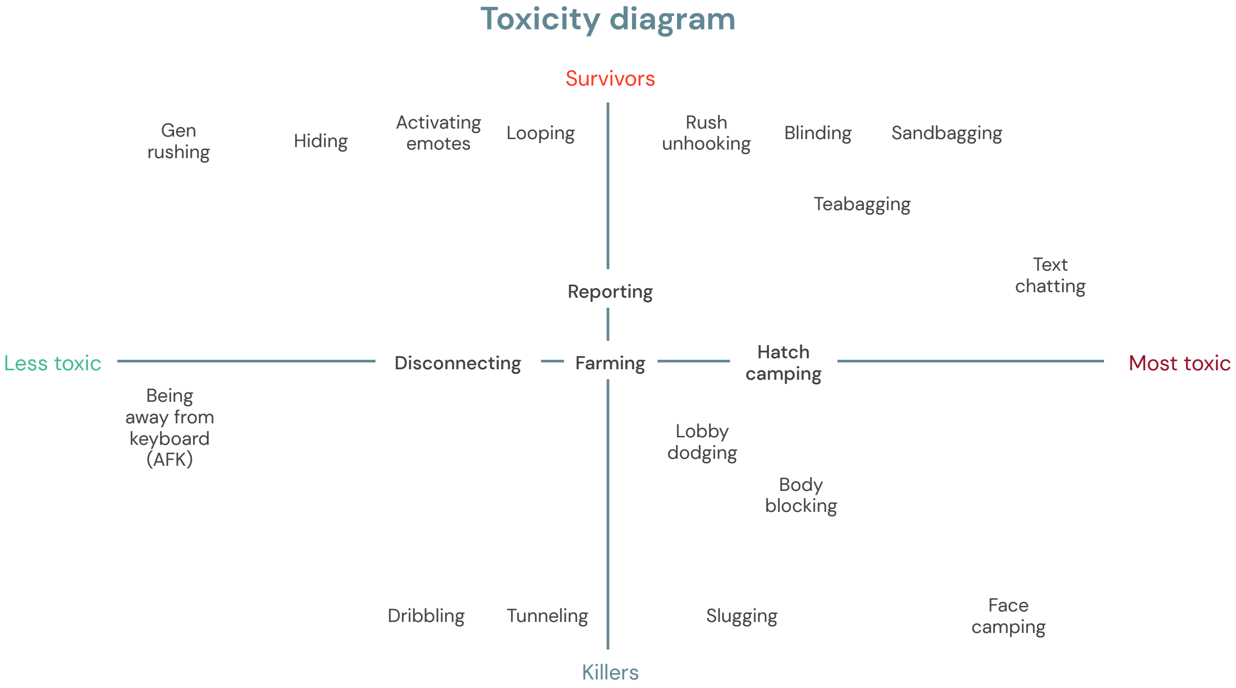


Figure 1
Matrix of toxic interactions that players experience

In addition to the personal toll that toxic behavior can have on gamers and the community — an issue that cannot be overstated — it is also damaging to the bottom line of many game studios. For example, a study from Michigan State University revealed that 80% of players recently experienced toxicity, and of those, 20% reported leaving the game due to these interactions. Similarly, a study from Tilburg University showed that having a disruptive or toxic encounter in the first session of the game led to players being over three times more likely to leave the game without returning. Given that player retention is a top priority for many studios, particularly as game delivery transitions from physical media releases to long-lived services, it's clear that toxicity must be curbed.

Compounding this issue related to churn, some companies face challenges related to toxicity early in development, even before launch. For example, Amazon's Crucible was released into testing without text or voice chat due in part to not having a system in place to monitor or manage toxic gamers and interactions. This illustrates that the scale of the gaming space has far surpassed most teams' ability to manage such behavior through reports or by intervening in disruptive interactions. Given this, it's essential for studios to integrate analytics into games early in the development lifecycle and then design for the ongoing management of toxic interactions.

Toxicity in gaming is clearly a multifaceted issue that has become a part of video game culture and cannot be addressed universally in a single way. That said, addressing toxicity within in-game chat can have a huge impact given the frequency of toxic behavior and the ability to automate detection of it using natural language processing (NLP).

Introducing the Toxicity Detection in Gaming Solution Accelerator from Databricks

Using toxic comment data from Jigsaw and Dota 2 game match data, this Solution Accelerator walks through the steps required to detect toxic comments in real time using NLP and your existing lakehouse. For NLP, this Solution Accelerator uses Spark NLP from John Snow Labs, an open source, enterprise-grade solution built natively on Apache Spark.™

The steps you will take in this Solution Accelerator are:

- Load the Jigsaw and Dota 2 data into tables using Delta Lake
- Classify toxic comments using multi-label classification (Spark NLP)
- Track experiments and register models using MLflow
- Apply inference on batch and streaming data
- Examine the impact of toxicity on game match data

Detecting toxicity within in-game chat in production

With this Solution Accelerator, you can now more easily integrate toxicity detection into your own games. For example, the reference architecture below shows how to take chat and game data from a variety of sources, such as streams, files, voice or operational databases, and leverage Databricks to ingest, store and curate data into feature tables for machine learning (ML) pipelines, in-game ML, BI tables for analysis and even direct interaction with tools used for community moderation.

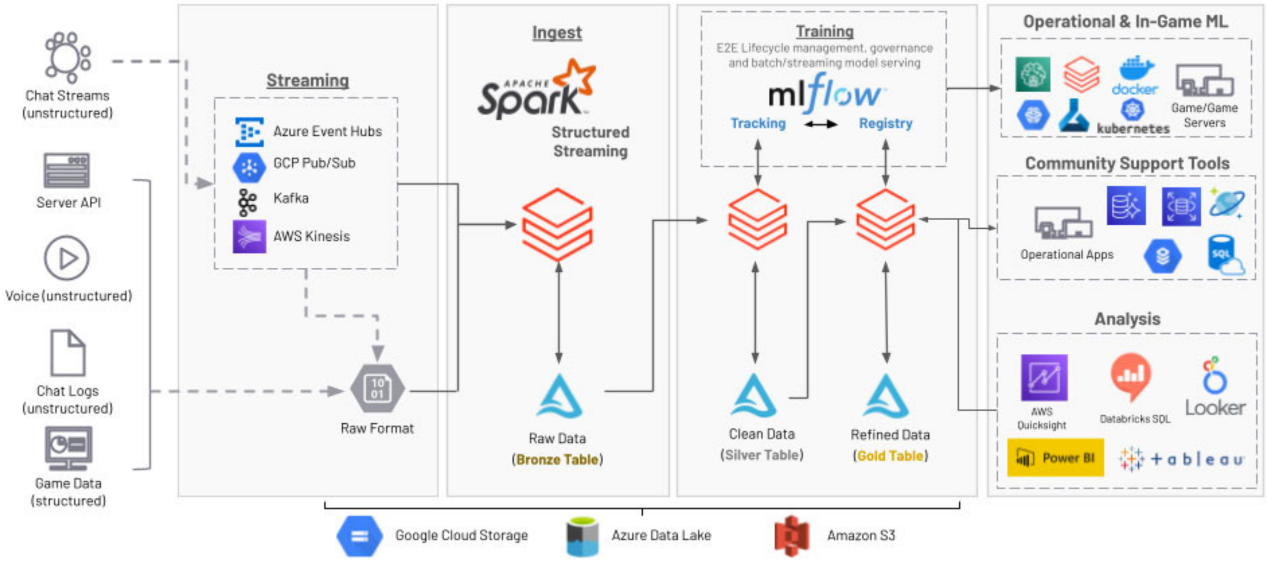


Figure 2
Toxicity detection reference architecture

Having a real-time, scalable architecture to detect toxicity in the community allows for the opportunity to simplify workflows for community relationship managers and the ability to filter millions of interactions into manageable workloads. Similarly, the possibility of alerting on severely toxic events in real time, or even automating a response such as muting players or alerting a CRM to the incident quickly, can have a direct impact on player retention. Likewise, having a platform capable of processing large data sets, from disparate sources, can be used to monitor brand perception through reports and dashboards.

Getting started

The goal of this Solution Accelerator is to help support the ongoing management of toxic interactions in online gaming by enabling real-time detection of toxic comments within in-game chat. Get started today by importing this Solution Accelerator directly into your Databricks workspace.

Once imported you will have notebooks with two pipelines ready to move to production.

- ML pipeline using multi-label classification with training on real-world English data sets from Google Jigsaw. The model will classify and label the forms of toxicity in text.
- Real-time streaming inference pipeline leveraging the toxicity model. The pipeline source can be easily modified to ingest chat data from all the common data sources.

With both of these pipelines, you can begin understanding and analyzing toxicity with minimal effort. This Solution Accelerator also provides a foundation to build, customize and improve the model with relevant data to game mechanics and communities.

Start experimenting with these free Databricks notebooks.

SECTION 2.7

Driving Transformation at Northwestern Mutual (Insights Platform) by Moving Toward a Scalable, Open Lakehouse Architecture

by MADHU KOTIAN

July 15, 2021

Digital transformation has been front and center in most contemporary big data corporate initiatives, especially in companies with a heavy legacy footprint. One of the underpinning components in digital transformation is data and its related data store. For 160+ years, Northwestern Mutual has been helping families and businesses achieve financial security. With over $31B in revenue, 4.6M+ clients and 9,300+ financial professionals, there are not too many companies that have this volume of data across a variety of sources.

Data ingestion is a challenge in this day and age, when organizations deal with millions of data points coming in different formats, time frames and from different directions at an unprecedented volume. We want to make data ready for analysis to make sense of it. Today, I am excited to share our novel approach to transforming and modernizing our data ingestion process, scheduling process and journey with data stores. One thing we learned is that an effective approach is multifaceted, which is why in addition to the technical arrangements I'll walk through our plan to onboard our team.

Challenges faced

Before we embarked on our transformation, we worked with our business partners to really understand our technical constraints and help us shape the problem statement for our business case.

The business pain point we identified was a lack of integrated data, with customer and business data coming from different internal and external teams and data sources. We realized the value of real-time data but had limited access to production/real-time data that could enable us to make business decisions in a timely manner. We also learned that data stores built by the business team resulted in data silos, which in turn caused data latency issues, increased cost of data management and unwarranted security constraints.

Furthermore, there were technical challenges with respect to our current state. With increased demand and additional data needed, we experienced constraints with infrastructure scalability, data latency, cost of managing data silos, data size and volume limitations, and data security issues. With these challenges mounting, we knew we had a lot to take on and needed to find the right partners to help us in our transformation journey.

Solution analysis

We needed to become data-driven to be competitive and serve our customers better and optimize internal processes. We explored various options and performed several POCs to select a final recommendation. The following were the must-haves for our go-forward strategy:

- An all-inclusive solution for our data ingestion, data management and analytical needs
- A modern data platform that can effectively support our developers and business analysts to perform their analysis using SQL
- A data engine that can support ACID transactions on top of S3 and enable role-based security
- A system that can effectively secure our PII/PHI information
- A platform that can automatically scale based on the data processing and analytical demand

Our legacy infrastructure was based on MSBI Stack. We used SSIS for ingestion, SQL Server for our data store, Azure Analysis Service for tabular model and Power BI for dashboarding and reporting. Although the platform met the needs of the business initially, we had challenges around scaling with increased data volume and data processing demand, and constrained our data analytical expectations. With additional data needs, our data latency issues from load delays and a data store for specific business needs caused data silos and data sprawl.

Security became a challenge due to the spread of data across multiple data stores. We had approximately 300 ETL jobs that took more than 7 hours from our daily jobs. The time to market for any change or new development was roughly 4 to 6 weeks (depending on the complexity).

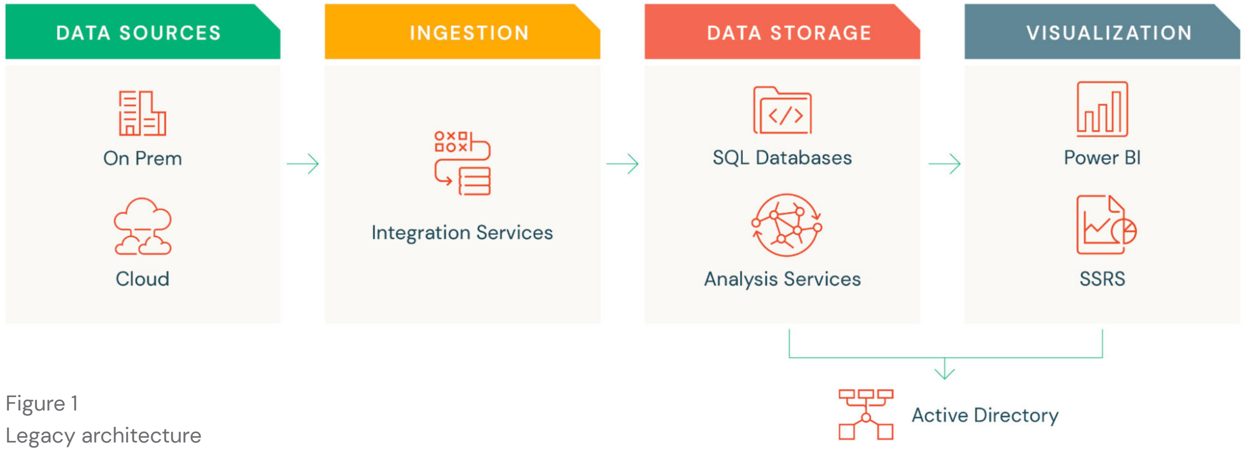


Figure 1
Legacy architecture

After evaluating multiple solutions in the marketplace, we decided to move forward with Databricks to help us deliver one integrated data management solution on an open lakehouse architecture.

Databricks being developed on top of Apache Spark™ enabled us to use Python to build our custom framework for data ingestion and metadata management. It provided us the flexibility to perform ad hoc analysis and other data discoveries using the notebook. Databricks Delta Lake (the storage layer built on top of our data lake) provided us the flexibility to implement various database management functions (ACID transactions, metadata governance, time travel, etc.) including the implementation of required security controls. Databricks took the headache out from managing/scaling the cluster and reacted effectively to the pent-up demand from our engineers and business users.

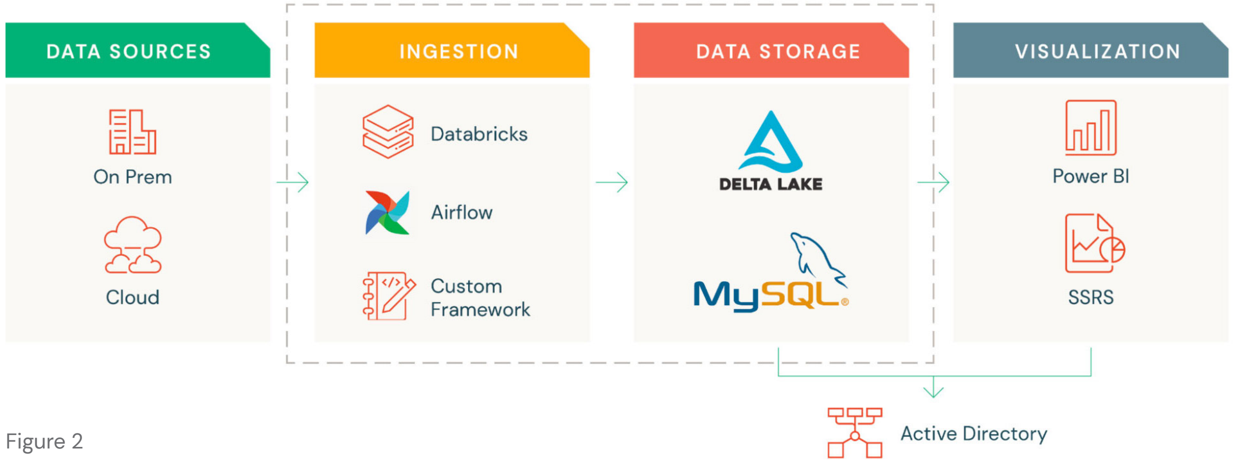


Figure 2
Architecture with Databricks

Migration approach and onboarding resources

We started with a small group of engineers and assigned them to a virtual team from our existing scrum team. Their goal was to execute different POC, build on the recommended solution, develop best practices and transition back to their respective team to help with the onboarding. Leveraging existing team members favored us better because they had existing legacy system knowledge, understood the current ingestion flow/business rules, and were well versed in at least one programming knowledge (data engineering + software engineering knowledge). This team first trained themselves in Python, understood intricate details of Spark and Delta, and closely partnered with the Databricks team to validate the solution/approach. While the team was working on forming the future state, the rest of our developers worked on delivering the business priorities.

Since most of the developers were MSBI Stack engineers, our plan of action was to deliver a data platform that would be frictionless for our developers, business users and our field advisors.

- We built an ingestion framework that covered all our data load and transformation needs. It had in-built security controls, which maintained all the metadata and secrets of our source systems. The ingestion process accepted a JSON file that included the source, target and required transformation. It allowed for both simple and complex transformation.
- For scheduling, we ended up using Airflow, but given the complexity of the DAG, we built our own custom framework on top of Airflow, which accepted a YAML file that included job information and its related interdependencies.
- For managing schema-level changes using Delta, we built our own custom framework which automated different database type operations (DDL) without requiring developers to have break-glass access to the data store. This also helped us to implement different audit controls on the data store.

In parallel, the team also worked with our security team to make sure we understood and met all the criteria for data security (encryption in transit, encryption at rest and column level encryption to protect PII information).

Once these frameworks were set up, the cohort team deployed an end-to-end flow (source to target with all transformation) and generated a new set of reports/dashboards on Power BI pointing to Delta Lake. The goal was to test the performance of our end-to-end process, validate the data and obtain any feedback from our field users. We incrementally improved the product based on the feedback and outcomes of our performance/validation test.

Simultaneously, we built training guides and how-tos to onboard our developers. Soon after, we decided to move the cohort team members to their respective teams while retaining a few to continue to support the platform infrastructure (DevOps). Each scrum team was responsible for managing and delivering their respective set of capabilities/features to the business. Once the team members moved back to their respective teams, they embarked on the task to adjust the velocity of the team to include the backlog for migration effort. The team leads were given specific guidance and appropriate goals to meet the migration goals for different Sprint/Program Increments. The team members who were in the cohort group were now the resident experts and they helped their team onboard to the new platform. They were available for any ad hoc questions or assistance.

As we incrementally built our new platform, we retained the old platform for validation and verification.

The beginning of success

The overall transformation took us roughly a year and a half, which is quite a feat given that we had to build all the frameworks, manage business priorities, manage security expectations, retool our team and migrate the platform. Overall load time came down remarkably from 7 hours to just 2 hours. Our time to market was about 1 to 2 weeks, down significantly from 4 to 6 weeks. This was a major improvement in which I know will extend itself to our business in several ways.

Our journey is not over. As we continue to enhance our platform, our next mission will be to expand on the lakehouse pattern. We are working on migrating our platform to E2 and deploying Databricks SQL. We are working on our strategy to provide a self-service platform to our business users to perform their ad hoc analysis and also enable them to bring their own data with an ability to perform analysis with our integrated data. What we learned is that we benefited greatly by using a platform that was open, unified and scalable. As our needs and capabilities grow, we know we have a robust partner in Databricks.

Hear more about Northwestern Mutual's journey to the Lakehouse

ABOUT MADHU KOTIAN

Madhu Kotian is the Vice President of Engineering (Investment Products Data, CRM, Apps and Reporting) at Northwestern Mutual. He has over 25+ years of experience in the field of Information Technology with experience and expertise in data engineering, people management, program management, architecture, design, development and maintenance using Agile practices. He is also an expert in data warehouse methodologies and implementation of data integration and analytics.

SECTION 2.8

How Databricks Data Team Built a Lakehouse Across Three Clouds and 50+ Regions

by JASON POHL and SURAJ ACHARYA

July 14, 2021

The internal logging infrastructure at Databricks has evolved over the years and we have learned a few lessons along the way about how to maintain a highly available log pipeline across multiple clouds and geographies. This blog will give you some insight as to how we collect and administer real-time metrics using our lakehouse platform, and how we leverage multiple clouds to help recover from public cloud outages.

When Databricks was founded, it only supported a single public cloud. Now, the service has grown to support the three major public clouds (AWS, Azure, GCP) in over 50 regions around the world. Each day, Databricks spins up millions of virtual machines on behalf of our customers. Our data platform team of less than 10 engineers is responsible for building and maintaining the logging telemetry infrastructure, which processes half a petabyte of data each day. The orchestration, monitoring and usage is captured via service logs that are processed by our infrastructure to provide timely and accurate metrics. Ultimately, this data is stored in our own petabyte-sized Delta Lake. Our Data Platform team uses Databricks to perform inter-cloud processing so that we can federate data where appropriate, mitigate recovery from a regional cloud outage and minimize disruption to our live infrastructure.

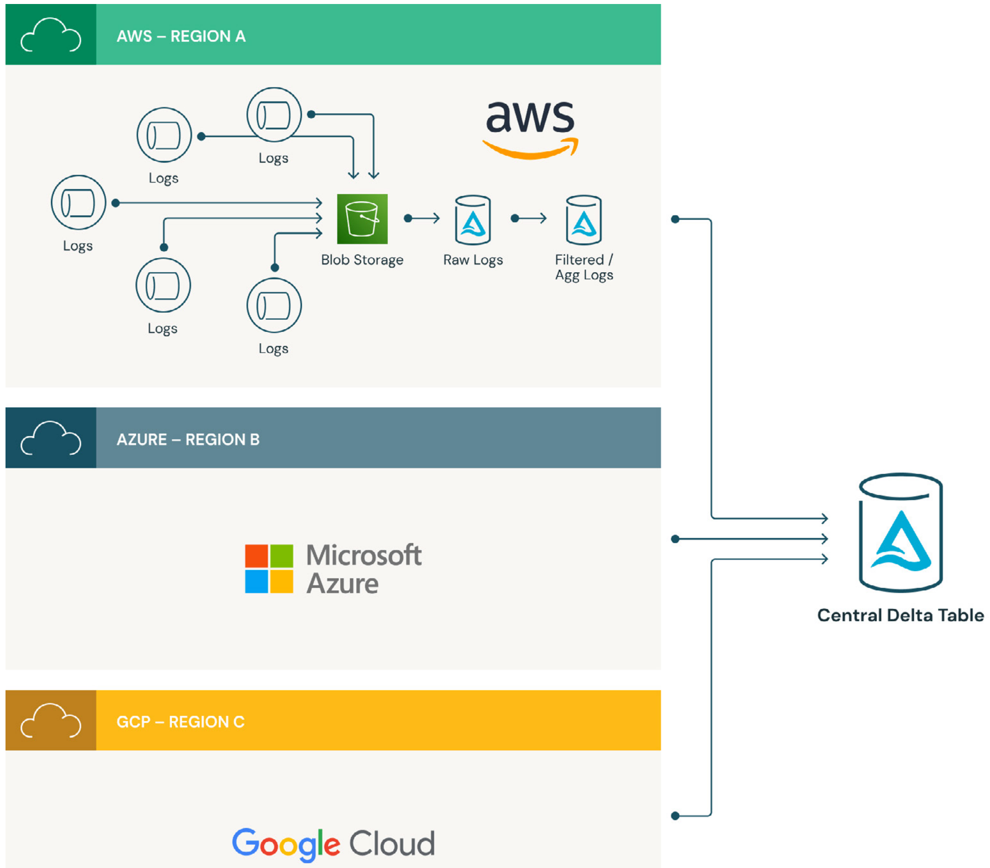


Figure 1

Pipeline architecture

Each cloud region contains its own infrastructure and data pipelines to capture, collect and persist log data into a regional Delta Lake. Product telemetry data is captured across the product and within our pipelines by the same process replicated across every cloud region. A log daemon captures the telemetry data and it then writes these logs onto a regional cloud storage bucket (S3, WASBS, GCS). From there, a scheduled pipeline will ingest the log files using Auto Loader (AWS | Azure | GCP), and write the data into a regional Delta table. A different pipeline will read data from the regional Delta table, filter it and write it to a centralized Delta table in a single cloud region.

Before Delta Lake

Prior to Delta Lake, we would write the source data to its own table in the centralized lake, and then create a view which was a union across all of those tables. This view needed to be calculated at runtime and became more inefficient as we added more regions:

```
CREATE OR REPLACE VIEW all_logs AS
SELECT * FROM (
  SELECT * FROM region_1.log_table
  UNION ALL
  SELECT * FROM region_2.log_table
  UNION ALL
  SELECT * FROM region_3.log_table
  ...
);
```

After Delta Lake

Today, we just have a single Delta Table that accepts concurrent write statements from over 50 different regions. While simultaneously handling queries against the data. It makes querying the central table as easy as:

```
SELECT * FROM central.all_logs;
```

The transactionality is handled by Delta Lake. We have deprecated the individual regional tables in our central Delta Lake and retired the UNION ALL view. The following code is a simplified representation of the syntax that is executed to load the data approved for egress from the regional Delta Lakes to the central Delta Lake:

```
spark.readStream.format(“delta”)
  .load(regional_source_path)
  .where(“egress_approved = true”)
  .writeStream
  .format(“delta”)
  .outputMode(“append”)
  .option(“checkpointLocation”, checkpoint_path)
  .start(central_target_path)
```

Disaster recovery

One of the benefits of operating an inter-cloud service is that we are well positioned for certain disaster recovery scenarios. Although rare, it is not unheard of for the compute service of a particular cloud region to experience an outage. When that happens, the cloud storage is accessible, but the ability to spin up new VMs is hindered. Because we have engineered our data pipeline code to accept configuration for the source and destination paths, this allows us to quickly deploy and run data pipelines in a different region to where the data is being stored. The cloud for which cloud the cluster is created in is irrelevant to which cloud the data is read or written to.

There are a few data sets which we safeguard against failure of the storage service by continuously replicating the data across cloud providers. This can easily be done by leveraging Delta deep clone functionality as described in this blog. Each time the clone command is run on a table, it updates the clone with only the incremental changes since the last time it was run. This is an efficient way to replicate data across regions and even clouds.

Minimizing disruption to live data pipelines

Our data pipelines are the lifeblood of our managed service and part of a global business that doesn't sleep. We can't afford to pause the pipelines for an extended period of time for maintenance, upgrades or backfilling of data. Recently, we needed to fork our pipelines to filter a subset of the data normally written to our main table to be written to a different public cloud. We were able to do this without disrupting business as usual.

By following these steps we were able to deploy changes to our architecture into our live system without causing disruption.

> First, we performed a deep clone of the main table to a new location on the other cloud. This copies both the data and the transaction log in a way to ensure consistency.
>
> Second, we released the new config to our pipelines so that the majority of data continues to be written to the central main table, and the subset of data writes to the new cloned table in the different cloud. This change can be made easily by just deploying a new config, and the tables receive updates for just the new changes they should receive.
>
> Next, we ran the same deep clone command again. Delta Lake will only capture and copy the incremental changes from the original main table to the new cloned table. This essentially backfills the new table with all the changes to the data between steps 1 and 2.
>
> Finally, the subset of data can be deleted from the main table and the majority of data can be deleted from the cloned table.
>
> Now both tables represent the data they are meant to contain, with full transactional history, and it was done live without disrupting the freshness of the pipelines.

Summary

Databricks abstracts away the details of individual cloud services whether that be for spinning up infrastructure with our cluster manager, ingesting data with Auto Loader, or performing transactional writes on cloud storage with Delta Lake. This provides us with an advantage in that we can use a single code-base to bridge the compute and storage across public clouds for both data federation and disaster recovery. This inter-cloud functionality gives us the flexibility to move the compute and storage wherever it serves us and our customers best.

SECTION

03 Customer Stories

Atlassian

ABN AMRO

J.B. Hunt

SECTION 3.1

Atlassian

Atlassian is a leading provider of collaboration, development and issue-tracking software for teams. With over 150,000 global customers (including 85 of the Fortune 100), Atlassian is advancing the power of collaboration with products including Jira, Confluence, Bitbucket, Trello and more.

USE CASE

Atlassian uses the Databricks Lakehouse Platform to democratize data across the enterprise and drive down operational costs. Atlassian currently has a number of use cases focused on putting the customer experience at the forefront.

Customer support and service experience
With the majority of their customers being server-based (using products like Jira and Confluence), Atlassian set out to move those customers into the cloud to leverage deeper insights that enrich the customer support experience.

Marketing personalization
The same insights could also be used to deliver personalized marketing emails to drive engagement with new features and products.

Anti-abuse and fraud detection
They can predict license abuse and fraudulent behavior through anomaly detection and predictive analytics.

At Atlassian, we need to ensure teams can collaborate well across functions to achieve constantly evolving goals. A simplified lakehouse architecture would empower us to ingest high volumes of user data and run the analytics necessary to better predict customer needs and improve the experience of our customers. A single, easy-to-use cloud analytics platform allows us to rapidly improve and build new collaboration tools based on actionable insights.

Rohan Dhupelia
Data Platform Senior Manager, Atlassian

SOLUTION AND BENEFITS

Atlassian is using the Databricks Lakehouse Platform to enable data democratization at scale, both internally and externally. They have moved from a data warehousing paradigm to standardization on Databricks, enabling the company to become more data driven across the organization. Over 3,000 internal users in areas ranging from HR and marketing to finance and R&D — more than half the organization — are accessing insights from the platform on a monthly basis via open technologies like Databricks SQL. Atlassian is also using the platform to drive more personalized support and service experiences to their customers.

- Delta Lake underpins a single lakehouse for PBs of data accessed by 3,000+ users across HR, marketing, finance, sales, support and R&D
- BI workloads powered by Databricks SQL enable dashboard reporting for more users
- MLflow streamlines MLOps for faster delivery
- Data platform unification eases governance, and self-managed clusters enable autonomy

With cloud-scale architecture, improved productivity through cross-team collaboration, and the ability to access all of their customer data for analytics and ML, the impact on Atlassian is projected to be immense. Already the company has:

- Reduced the cost of IT operations (specifically compute costs) by 60% through moving 50,000+ Spark jobs from EMR to Databricks with minimal effort and low-code change
- Decreased delivery time by 30% with shorter dev cycles
- Reduced data team dependencies by 70% with more self-service enabled throughout the organization

Learn More

SECTION 3.2

ABN AMRO

As an established bank, ABN AMRO wanted to modernize their business but were hamstrung by legacy infrastructure and data warehouses that complicated access to data across various sources and created inefficient data processes and workflows. Today, Azure Databricks empowers ABN AMRO to democratize data and AI for a team of 500+ empowered engineers, scientists and analysts who work collaboratively on improving business operations and introducing new go-to-market capabilities across the company.

USE CASE

ABN AMRO uses the Databricks Lakehouse Platform to deliver financial services transformation on a global scale, providing automation and insight across operations.

Personalized finance
ABN AMRO leverages real-time data and customer insights to provide products and services tailored to customers' needs. For example, they use machine learning to power targeted messaging within their automated marketing campaigns to help drive engagement and conversion.

Risk management
Using data-driven decision-making, they are focused on mitigating risk for both the company and their customers. For example, they generate reports and dashboards that internal decision makers and leaders use to better understand risk and keep it from impacting ABN AMRO's business.

Fraud detection
With the goal of preventing malicious activity, they're using predictive analytics to identify fraud before it impacts their customers. Among the activities they're trying to address are money laundering and fake credit card applications.

> **Databricks has changed the way we do business. It has put us in a better position to succeed in our data and AI transformation as a company by enabling data professionals with advanced data capabilities in a controlled and scalable way.**
>
> **Stefan Groot**
> Head of Analytics Engineering,
> ABN AMRO

SOLUTION AND BENEFITS

Today, Azure Databricks empowers ABN AMRO to democratize data and AI for a team of 500+ engineers, scientists and analysts who work collaboratively on improving business operations and introducing new go-to-market capabilities across the company.

- Delta Lake enables fast and reliable data pipelines to feed accurate and complete data for downstream analytics
- Integration with Power BI enables easy SQL analytics and feeds insights to 500+ business users through reports and dashboards
- MLflow speeds deployment of new models that improve the customer experience — with new use cases delivered in under two months

10x faster
time to market — use cases deployed in two months

100+
use cases to be delivered over the coming year

500+
empowered business and IT users

Learn More

SECTION 3.2

J.B. HUNT

"

What Databricks has really given us is a foundation for the most innovative digital freight marketplace by enabling us to leverage AI to deliver the best carrier experience possible.

Joe Spinelle
Director, Engineering and Technology,
J.B. Hunt

In their mission to build North America's most efficient digital transportation network, J.B. Hunt wanted to streamline freight logistics and provide the best carrier experience — but legacy architecture, their lack of AI capabilities and the inability to securely handle big data caused significant roadblocks. However, after implementing the Databricks Lakehouse Platform and Immuta, J.B. Hunt is now able to deliver operational solutions that range from improving supply chain efficiencies to boosting driver productivity — resulting in significant IT infrastructure savings and revenue gains.

USE CASE

J.B. Hunt uses Databricks to deliver industry-leading freight carrier analytics via their Carrier 360 platform, driving down costs while increasing driver productivity and safety. Use cases include freight logistics, customer 360, personalization and many more.

SOLUTION AND BENEFITS

J.B. Hunt uses the Databricks Lakehouse Platform to build North America's most secure and efficient freight marketplace — streamlining logistics, optimizing carrier experiences and cutting costs.

- Delta Lake federates and democratizes data for real-time route optimizations and driver recommendations via the Carrier 360 platform
- Notebooks boost data team productivity to deliver more use cases faster
- MLflow speeds deployment of new models that improve driver experience

$2.7M
in IT infrastructure savings, increasing profitability

5%
increase in revenue driven by improved logistics

99.8% faster
recommendations for a better carrier experience

Learn More

databricks

About Databricks

Databricks is the data and AI company. More than 5,000 organizations worldwide — including Comcast, Condé Nast, H&M and over 40% of the Fortune 500 — rely on the Databricks Lakehouse Platform to unify their data, analytics and AI. Databricks is headquartered in San Francisco, with offices around the globe. Founded by the original creators of Apache Spark,™ Delta Lake and MLflow, Databricks is on a mission to help data teams solve the world's toughest problems. To learn more, follow Databricks on Twitter, LinkedIn and Facebook.

START YOUR FREE TRIAL

Contact us for a personalized demo
databricks.com/contact

databricks