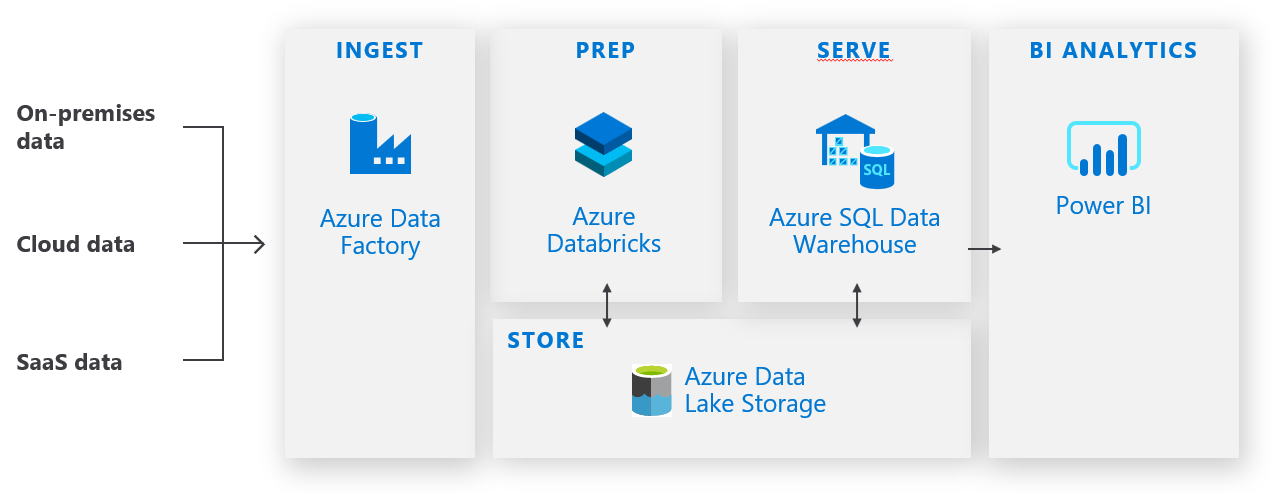


Modern Data Warehouse Pattern

# Modern Data Warehouse Pattern

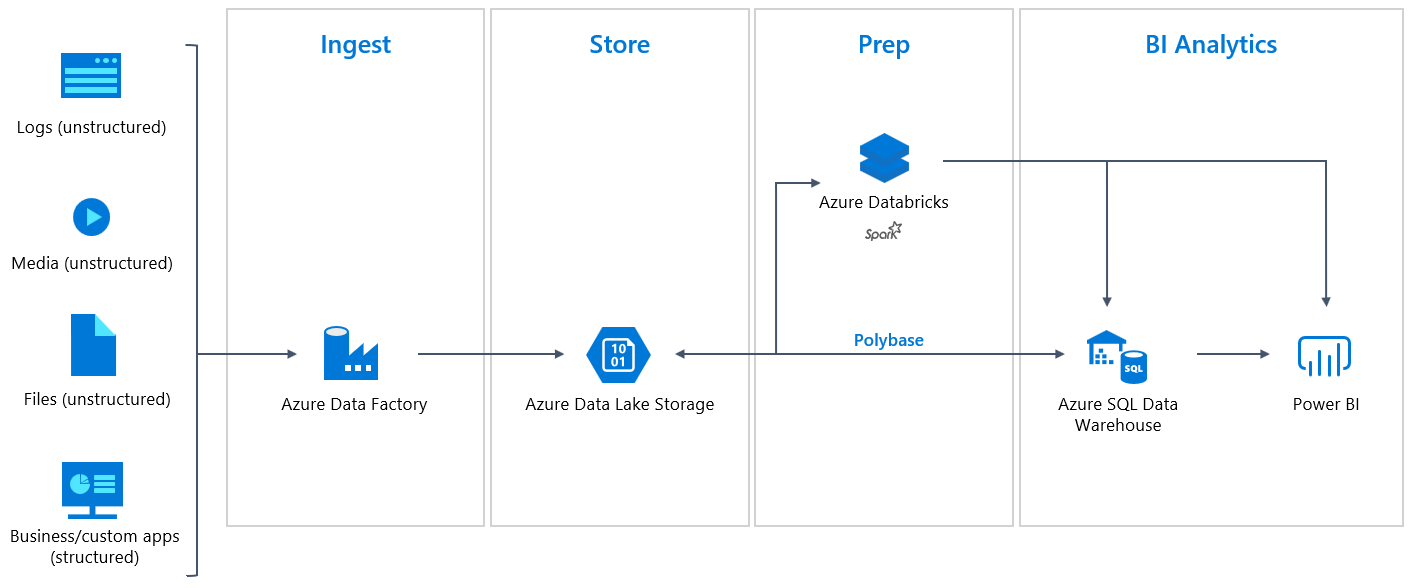


The [Modern Data Warehouse Pattern](https://azure.microsoft.com/en-us/solutions/architecture/modern-data-warehouse/) is a solution pattern designed to achieve high performance BI Analytics on Big Data. Many customers use this pattern for success when modernizing their Data Warehouse on Azure. One recent customer example is a retailer, [Columbia Sportswear](https://www.columbia.com/). They came from an on-prem data warehouse powered by TerraData and Informatica. They had siloed data, slow time to insights, and their volume of data was growing exponentially. In 2 months, they migrated over 750 informatica ETL pipelines to Azure Databricks + SQL DW orchestrated by ADF. This modernization resulted in 50% overall cost savings and 70% reduction in time to insights.

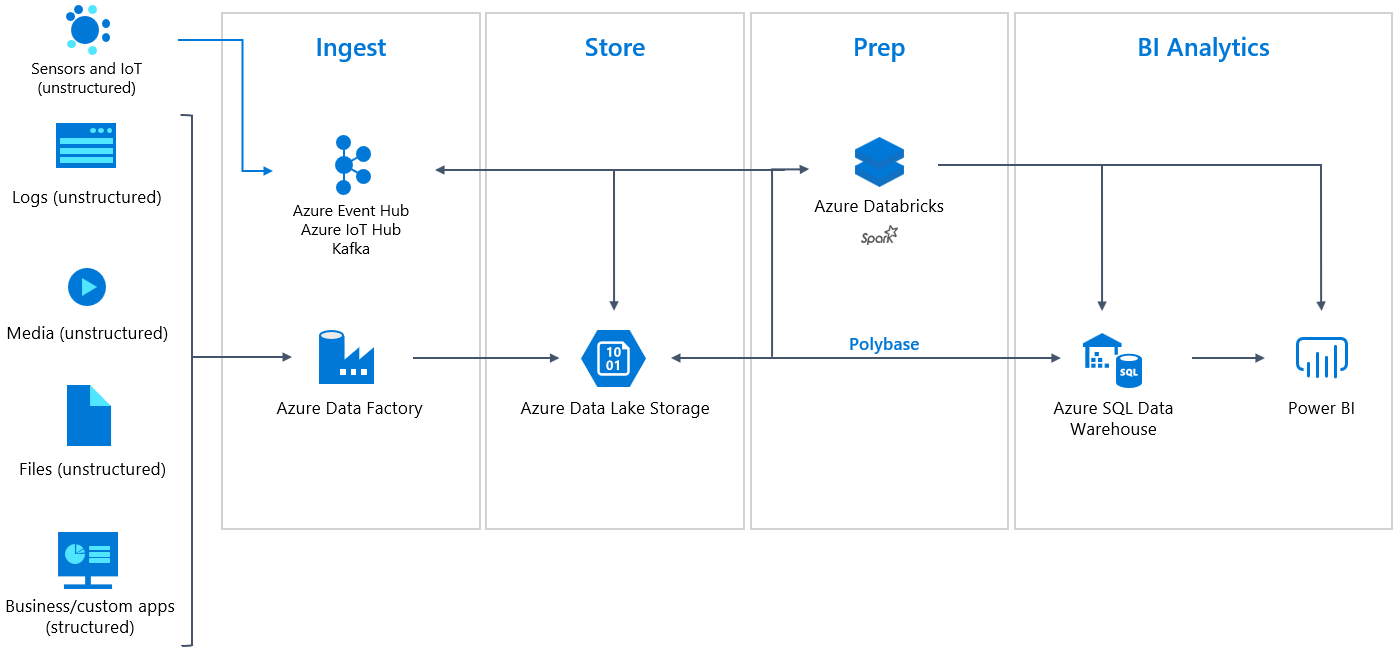
# How Azure Databricks helps realize the Modern Data Warehouse

# Recommended Architectures

**Traditional Batch ETL workloads**

Azure Databricks is well suited for [traditional batch ETL workloads](https://docs.microsoft.com/en-us/azure/azure-databricks/databricks-extract-load-sql-data-warehouse). You can work with structured or unstructured data, raw media, or other complex data formats. With Apache Spark high performance distributed compute you can efficiently query over, transform, join, enrich, and process all your data in your Data Lake. Save this enriched and cleaned data back to your Data Lake or stage it in SQL Data Warehouse for scalable serving to PBI for BI Analytics.

**Batch and Streaming Workloads**



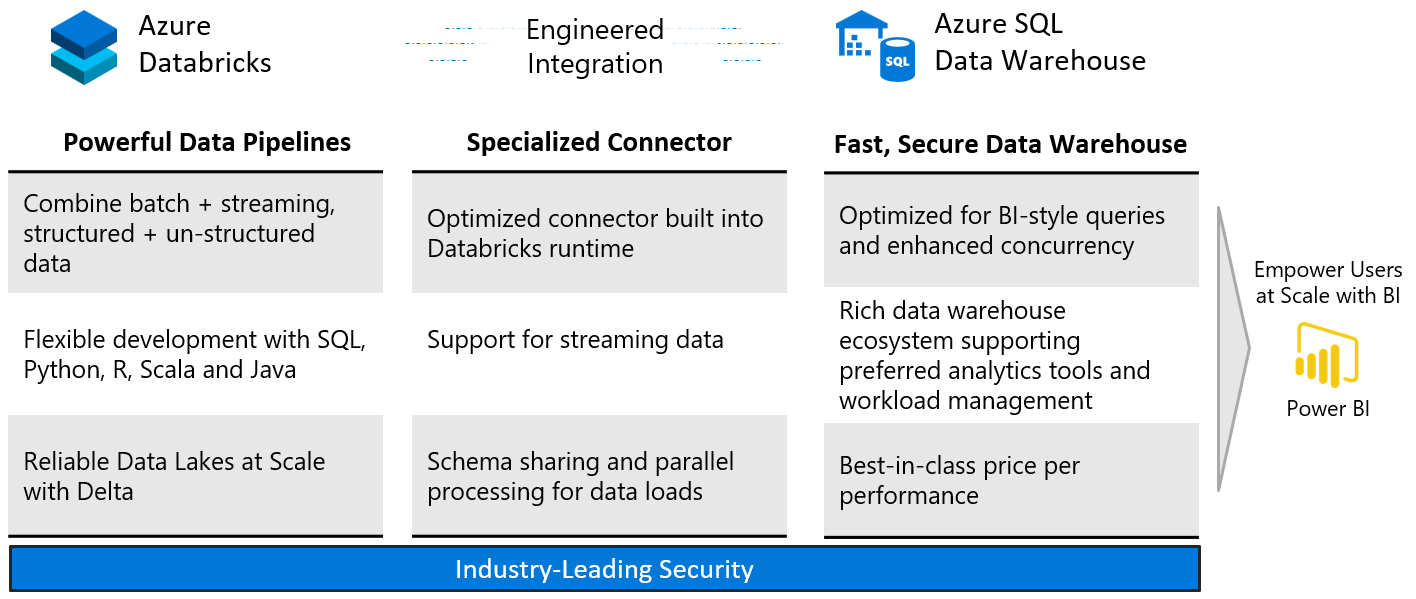
Azure Databricks’ [friendly structured streaming APIs](https://docs.azuredatabricks.net/getting-started/spark/streaming.html) make it simple and efficient to combine Batch and Streaming data efficiently. You can easily connect to streaming sources such as Apache Kafka, Event Hubs, IoT Hubs. With multi stream and batch joins, advanced windowing functions, exactly once semantics, fault-tolerant checkpointing, and [streaming support to SQL Data Warehouse](https://docs.azuredatabricks.net/spark/latest/data-sources/azure/sql-data-warehouse.html#streaming-support) you can [significantly reduce your time to insights](https://databricks.com/blog/2017/06/06/simple-super-fast-streaming-engine-apache-spark.html) and update your BI Analytics in near-real-time. With the power of [Azure Databricks Managed Delta Lake](#_Azure_Databricks_-) (described in detail below), you can simplify CDC and lambda architectures, obtain ACID transactional guarantees, and achieve up to 100x performance improvements.

**Machine Learning Enrichment**

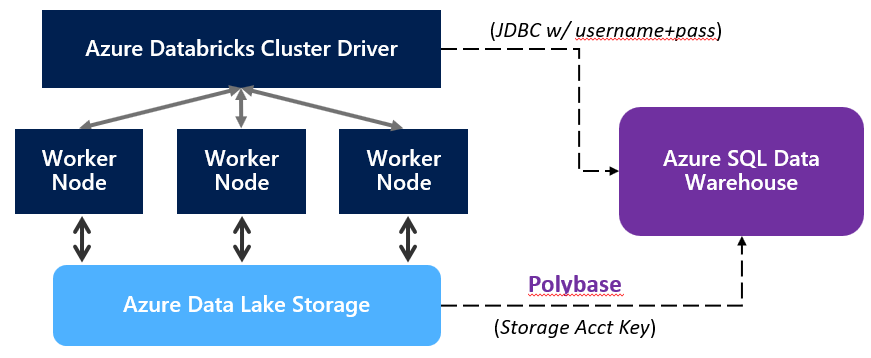
# 

With Azure Databricks you can [import popular Data Science libraries](https://docs.azuredatabricks.net/user-guide/libraries.html) to enrich your data with predictive analytics. Leverage [Azure Machine Learning SDKs together with MLFlow](https://docs.microsoft.com/en-us/azure/machine-learning/service/how-to-use-mlflow) to manage the complete ML Lifecycle from preparation, training, scoring, deploying machine learning models. When [you link your Azure ML workspace](https://docs.microsoft.com/en-us/azure/machine-learning/service/how-to-use-mlflow#set-mlflow-tracking-uri) with Azure Databricks you can take advantage of Azure ML experimentation run history, ML model management and drift tracking, model deployment and monitoring. The new [integration of Azure ML and MLFlow](https://azure.microsoft.com/en-us/blog/make-your-data-science-workflow-efficient-and-reproducible-with-mlflow/) announced at Spark Summit allows you to leverage AML when using MLFlow APIs.

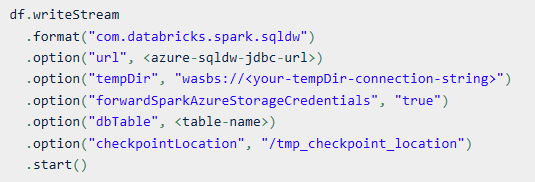
# Azure Databricks + SQL Datawarehouse Integration

**Better Together Story**

Azure Databricks has a tight integration with Azure SQL Data Warehouse. Together we developed a [high-performance connector](https://docs.azuredatabricks.net/spark/latest/data-sources/azure/sql-data-warehouse.html) that we built into the Azure Databricks runtime. This connector provides up to [80x performance improvement](https://www.linkedin.com/pulse/using-polybase-loading-data-from-databricks-azure-dw-high-speed-ma/) for loading data over the standard JDBC connections.



As shown in the diagram above, the Azure Databricks cluster driver [authenticates with Azure SQL Data Warehouse over JDBC](https://docs.azuredatabricks.net/spark/latest/data-sources/azure/sql-data-warehouse.html#authentication). Then each worker node in parallel begins loading data in a temporary storage location, where Polybase does high performance loads.



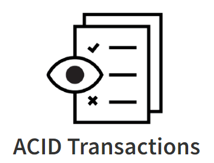
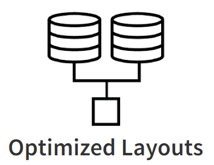
In the [sample code](https://docs.azuredatabricks.net/spark/latest/data-sources/azure/sql-data-warehouse.html#usage-batch) above you can see how you: (full documentation)

1. Specify the native connector built into the runtime
2. Pass a jdbc connection and a temp ADLS directory
3. You can forward the storage credentials (which are encrypted)
4. Specify the table you are writing to and a fault-tolerant checkpoint location.

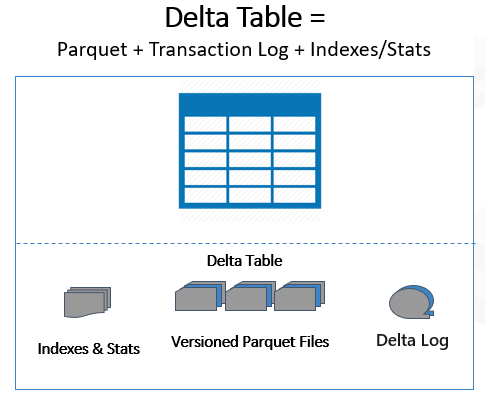
# Azure Databricks - Delta Lake

Delta Lake was released GA in Feb 2019, and announced as an open source Apache project in April 2019.

[Delta Lake](https://delta.io/) makes your Data Lakes reliable at scale with up to 100x performance improvements:



The secret sauce behind Delta is a new storage format. (*Remember Azure Databricks is only compute and is never a storage layer, so this new storage format, Delta is stored on Azure Data Lake Storage.*)



Data Lakes typically have multiple pipelines reading and writing the same data concurrently. Data engineers have to go through a tedious process to ensure data integrity, due to the lack of transactions.

With Delta, we can version your parquet files, keep a transaction log and build advanced indexes and statistics on your data. This enables snapshot isolation for multiple concurrent read/writes, immediate rollback capabilities, and [ACID transactional guarantees](https://docs.azuredatabricks.net/delta/concurrency-control.html).

In typical data lakes, over time your data can start to be scattered across many small files leading to performance degradation. Data engineers again have a costly and complex process to compact and clean up these directories for performance tuning.

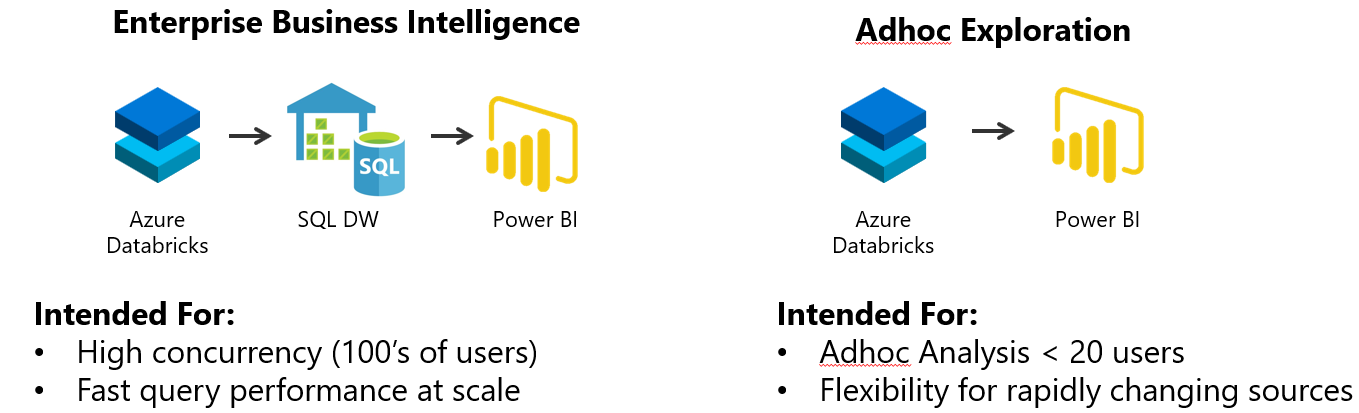
Delta provides an [OPTIMIZE command](https://docs.azuredatabricks.net/delta/optimizations.html) which is recommended to run on a schedule in a separate cluster which will automatically compact and clean small files. Following [best practices here](https://docs.azuredatabricks.net/delta/delta-on-databricks.html) you can achieve up to 100x faster query performance.

Many customers have adopted Delta and are transforming their Data Lakes into Reliable and High-Performance Delta Lakes. One customer [shared their story](https://www.youtube.com/watch?v=SFeBJxI4Q98). Their original data pipeline took 20 engineers, 24 weeks to build, and they could analyze 2 weeks of data with hours of delay. With Databricks Delta they built an equivalent data pipeline with 2 engineers in 2 weeks that could analyze 2 years of data with minutes of delay.

Azure Databricks Managed Delta Lake Guide: <https://docs.azuredatabricks.net/delta/index.html>  
OSS Github: <https://github.com/delta-io/delta>

# Product Positioning Recap

Even with the amazing features of Delta, Azure Databricks is not a replacement for a Data Warehouse. Azure Databricks with Delta on its own is not a successful pattern for enterprise grade BI analytics.



# Connecting Power BI directly to Azure Databricks is a functional and valid scenario for Adhoc exploration. But it will not match the concurrency scale and performance of other BI serving layers like Azure SQL Datawarehouse.

When following Modern Data Warehouse solution pattern, you are set up for successful and scalable Enterprise Grade BI Analytics.

