Last updated on ***Nov 11, 2025***

Lakeflow Spark Declarative Pipelines concepts

Learn what Lakeflow Spark Declarative Pipelines (SDP) is, the core concepts (such as pipelines, streaming tables, and materialized views) that define it, the relationships between those concepts, and the benefits of using it in your data processing workflows.

What is SDP?

Lakeflow Spark Declarative Pipelines is a declarative framework for developing and running batch and streaming data pipelines in SQL and Python. Lakeflow SDP extends and is interoperable with Apache Spark Declarative Pipelines, while running on the performance-optimized Databricks Runtime, and the Lakeflow Spark Declarative Pipelines `flows` API uses the same DataFrame API as Apache Spark and Structured Streaming. Common use cases for SDP include incremental data ingestion from sources such as cloud storage (including Amazon S3, Azure ADLS Gen2, and Google Cloud Storage) and message buses (such as Apache Kafka, Amazon Kinesis, Google Pub/Sub, Azure EventHub, and Apache Pulsar), incremental batch and streaming transformations with stateless and stateful operators, and real-time stream processing between transactional stores like message buses and databases.

For more details on declarative data processing, see Procedural vs. declarative data processing in Databricks.

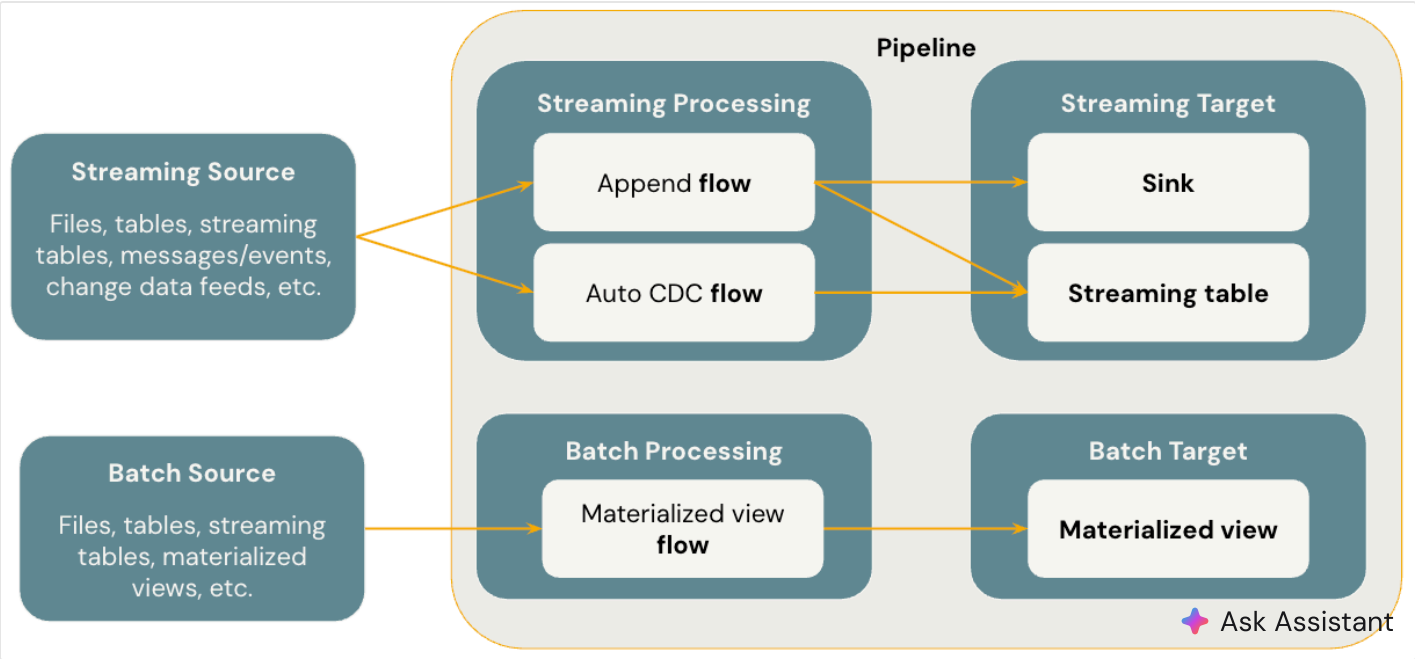
What are the benefits of SDP?

The declarative nature of SDP provides the following benefits compared to developing data processes with the Apache Spark and Spark Structured Streaming APIs and running them with the Databricks Runtime using manual orchestration via Lakeflow Jobs.

- **Automatic orchestration**: SDP orchestrates processing steps (called "flows") automatically to ensure the correct order of execution and the maximum level of parallelism for optimal performance. Additionally, pipelines automatically and efficiently retry transient failures. The retry process begins with the most granular and cost-effective unit: the Spark task. If the task-level retry fails, SDP proceeds to retry the flow, and then finally the entire pipeline if necessary.
- **Declarative processing**: SDP provides declarative functions that can reduce hundreds or even thousands lines of manual Spark and Structured Streaming code to only a few lines. The SDP AUTO CDC API simplifies processing of Change Data Capture (CDC) events with support for both SCD Type 1 and SCD Type 2. It eliminates the need for manual code to handle out-of-order events, and it does not require an understanding of streaming semantics or concepts like watermarks.
- **Incremental processing**: SDP provides an incremental processing engine for materialized views. To use it, you write your transformation logic with batch semantics, and the engine will only process new data and changes in the data sources whenever possible. Incremental processing reduces inefficient reprocessing when new data or changes occur in the sources and eliminates the need for manual code to handle incremental processing.

Key Concepts

The diagram below illustrates the most important concepts of Lakeflow Spark Declarative Pipelines.



Flows

A flow is the foundational data processing concept in SDP which supports both streaming and batch semantics. A flow reads data from a source, applies user-defined processing logic, and writes the result into a target. SDP shares the same streaming flow type (*Append*, *Update*, *Complete*) as Spark Structured Streaming. (Currently, only the *Append* flow is exposed.) For more details, see output modes in Structured Streaming.

Lakeflow Spark Declarative Pipelines also provides additional flow types:

- *AUTO CDC* is a unique streaming flow in Lakeflow SDP that handles out of order CDC events and supports both SCD Type 1 and SCD Type 2. Auto CDC is not available in Apache Spark Declarative Pipelines.
- *Materialized view* is a batch flow in SDP that only processes new data and changes in the source tables whenever possible.

For more details, see:

- Load and process data incrementally with SDP flows

Streaming tables

A *streaming table* is a form of Unity Catalog managed table that is also a streaming target for Lakeflow SDP. A streaming table can have one or more streaming flows (*Append*, *AUTO CDC*) written into it. *AUTO CDC* is a unique streaming flow that's only available to streaming tables in Databricks. You can define streaming flows explicitly and separately from their target streaming table. You can also define streaming flows implicitly as part of a streaming table definition.

For more details, see:

- How streaming tables work

Materialized views

A *materialized view* is also a form of Unity Catalog managed table and is a batch target. A materialized view can have one or more materialized view flows written into it. Materialized

views differ from streaming tables in that you always define the flows implicitly as part of the materialized view definition.

For more details, see:

- How materialized views work

Sinks

A *sink* is a streaming target for a pipeline and supports Delta tables, Apache Kafka topics, Azure EventHubs topics, and custom Python data sources. A sink can have one or more streaming flows (*Append*) written into it.

For more details, see:

- Stream records to external services with sinks

Pipelines

A *pipeline* is the unit of development and execution in Lakeflow Spark Declarative Pipelines. A pipeline can contain one or more flows, streaming tables, materialized views, and sinks. You use SDP by defining flows, streaming tables, materialized views, and sinks in your pipeline source code and then running the pipeline. While your pipeline runs, it analyzes the dependencies of your defined flows, streaming tables, materialized views, and sinks, and orchestrates their order of execution and parallelization automatically.

For more details, see:

- Configure a pipeline

Databricks SQL pipelines

Streaming tables and materialized views are two foundational capabilities in Databricks SQL. You can use standard SQL to create and refresh streaming tables and materialized views in Databricks SQL. Streaming tables and materialized views in Databricks SQL run on the same Databricks infrastructure and have the same processing semantics as they do in Lakeflow Spark Declarative Pipelines. When you use streaming tables and materialized views in

Databricks SQL, flows are defined implicitly as part of the streaming tables and materialized views definition.

For more details, see:

- Use pipelines in Databricks SQL

More information

- Tutorial: Build an ETL pipeline using change data capture
- Develop Lakeflow Spark Declarative Pipelines
- Pipeline Limitations
- Pipeline developer reference