EBOOK

Data Engineers' Handbook for Snowflake

4 Cloud Design Patterns for Data Ingestion and Transformation in Snowflake

www.streamsets.com

Table of Contents

StreamSets

Introduction

Snowflake Data Cloud adoption is accelerating with use cases spanning basic reporting, advanced analytics, operational insight, and data sharing. The diversity of these use cases, mounting requests, and new integrations make it difficult to quickly provide the analytics your team needs to make data-driven business decisions.

As a Data Engineer, you are left relying on a wide variety of approaches, hand coding or simple one-pattern/ecosystem tools to support your integrations.

StreamSets offers a single experience for all design patterns. In addition, it provides powerful developer extensibility with pre-built processors and custom expressors. There is further extensibility with Snowpark, enabling complex transformations on your data inside Snowflake. Plus, automatic, patented data drift capabilities spot changes in any data or upstream systems.

StreamSets

snowflake®

Critical Design Patterns for the Cloud

To successfully migrate data and data workloads to your Data Cloud platform there are these four common data pipeline design patterns:

- Ingesting to Data Cloud Platform
- Change Data Capture from Legacy to Data Cloud Platform
- Streaming Files into Snowflake Data Cloud using Kafka
- Native ELT on Snowflake Data Cloud with Snowpark

These four data pipeline patterns are the building blocks for ingesting, migrating, and transforming your data into Data Cloud platforms. Together, they help data engineers accelerate and simplify the move to the cloud supporting next-generation data analytics.

This handbook will walk you through the step-by-step process of building each of these critical design patterns. We provide multiple pipeline examples, best practices, design considerations, and use case examples. We will also explore what happens when something changes and how to create data pipelines that are resilient to change.

Finally, we consider the deployment and ongoing operations involved with running data pipelines that deliver continuous data. All workloads can be managed and optimized through interactive maps called topologies, from batch ingestion to change data capture to real-time streaming.

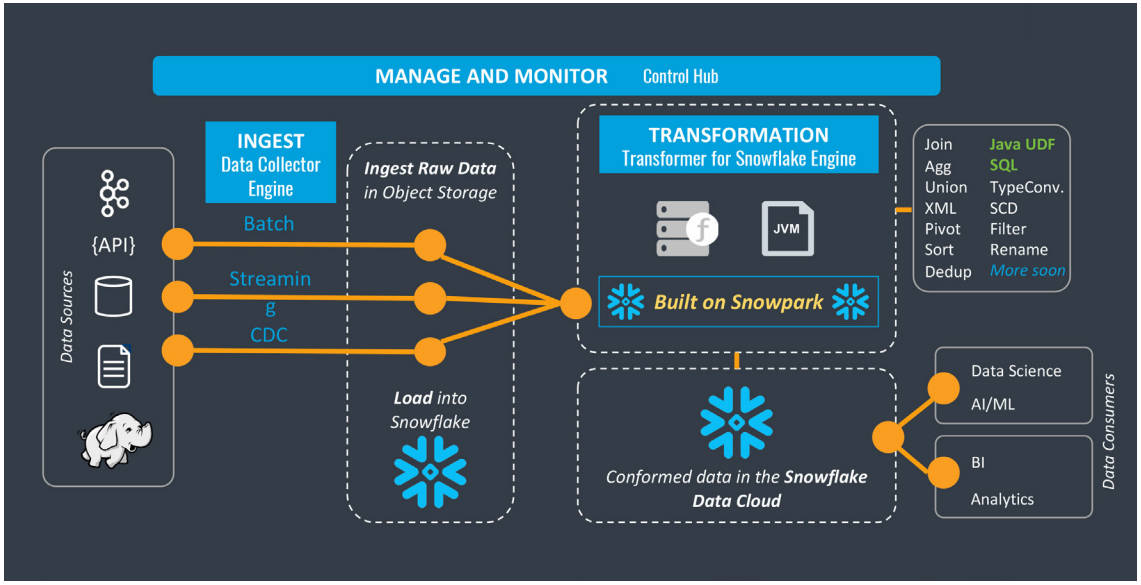

Figure 1. Design Patterns for Snowflake Data Cloud

The Role of the Data Engineer

The data engineer is the technical professional who understands how data analysts and data scientists need data, then builds the data pipeline(s) to deliver the right data, in the right format, to the right place. The best data engineers can anticipate the needs of the business, track the rise of new technologies, and maintain a complex and evolving data infrastructure.

But data engineers face many challenges as organizations evolve their use of data beyond traditional reporting to data science, AI, and machine learning. First, the project backlog is stressed and growing, putting pressure on the data engineering team. More data scientists and more data analysts mean more projects and demands for support from the data engineer.

Second, changes to data are accelerating in small and large ways. We call this "data drift": the unexpected and undocumented changes to data structure, semantics, and infrastructure resulting from modern data architectures. Keeping up with data drift creates a huge burden on data engineers and platform operators to keep the lights on and ensure there are no disruptions to the analytics delivery.

Third, as data platforms evolve, for example, from on-premises data lakes and EDW's into Data Cloud platforms, data engineers are on task for huge replatforming projects while still juggling their daily responsibilities.

Data engineers have many options, ranging from traditional ETL tools to simple ingest services to hand coding using a variety of programming languages. But juggling different design interfaces makes life hard for the data engineer. Why is that? They have to choose between powerful tools that require specialized skills or black box utilities for easy data ingest pipelines that are painful to maintain and operate continuously.

In addition, these approaches lead to brittle mappings or pipelines that require significant rework every time anything changes in the source or destination. Engineers can end up spending 80% of their time on maintenance, leaving very little time for new, value-added work.

This handbook outlines 4 data pipelines that can be implemented as "smart" data pipelines so you can go fast to get the data to the business and be confident that the pipelines you're building will hold up for ongoing operations.

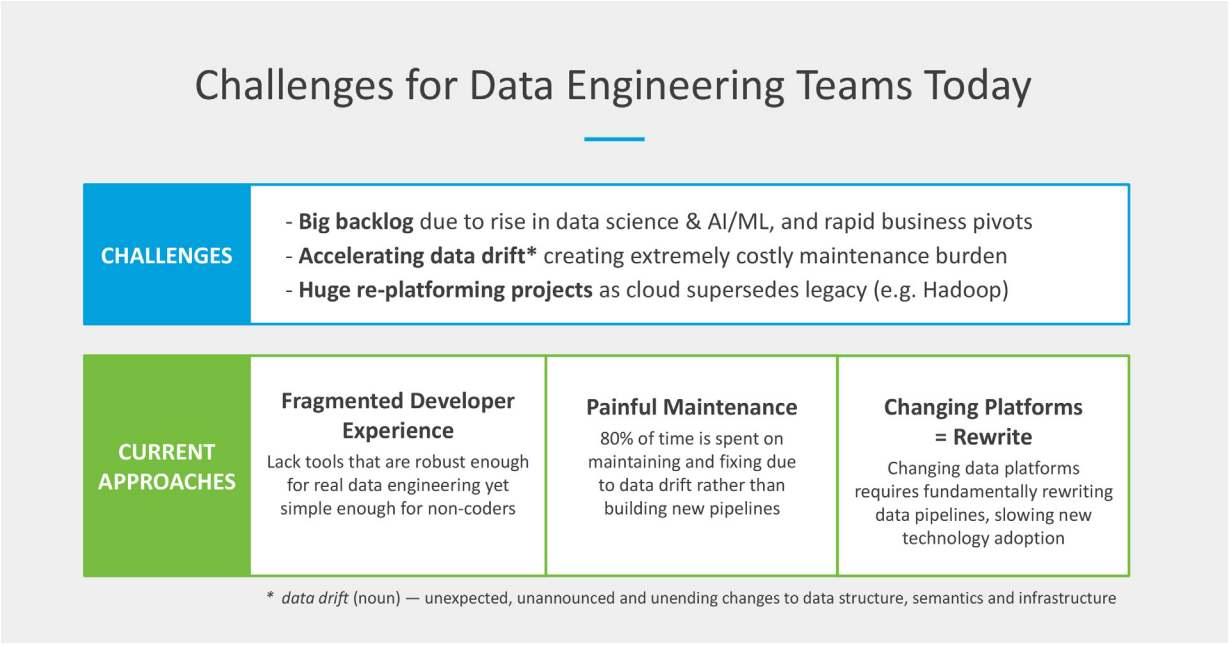

Figure 2. Modern Challenges for Data Engineers

The Rise of Smart Data Pipelines

A smart data pipeline is a data pipeline that is designed to operate continuously with as little manual intervention as possible. Smart data pipelines are essential in highly dynamic Data Cloud environments where data flows from multiple data platforms, both on-premises and cloud, and where data drift is everywhere.

What makes a data pipeline smart?

- ☑ Smart data pipelines use intent-driven design to abstract away the "how" of implementation from the "what" so engineers can focus on the business meaning and logic of the data.
- ☑ Smart data pipelines expect and are resilient to data drift.
- ☑ Smart data pipelines ensure portability across different platforms and clouds.

As we present each of the four design patterns essential for migrating to data clouds, we look at the difference smart data pipelines make and how they adapt to change.

The Snowflake Data Cloud

Companies in every industry acknowledge that data is one of their most important assets. However, companies are falling short of realizing the potential of data because of the proliferation of data silos. They are expensive and time-consuming to extract value from, and governance and collaboration are nearly impossible across multiple technologies and clouds.

The Data Cloud is one global, unified system connecting companies and data providers to the most relevant data for their business. **The Data Cloud enables 3 key functions: access to data, governance, and actionable functions.** Leading organizations across every industry run in the Data Cloud and fuel the global network as they share and collaborate in new ways. The Data Cloud is a new breed of data platform that brings the added advantage of cost-effectiveness and scalability with pay-as-you-go pricing models, a serverless approach, and on-demand resources. This is made possible by separating compute and storage to provide a layer specifically for fast analytics, reporting, and data mining.

StreamSets and Snowflake are Partnered to Accelerate Reliable Data integration to the Data Cloud

4 Design Patterns for Your Data Cloud

Pipeline Example #1: Ingest to Data Cloud Platform

Pipeline Example #2: Change Data Capture from Legacy to Data Cloud Platform

Pipeline Example #3: Streaming Files into Snowflake Data Cloud using Apache Kafka

Pipeline Example #4: Native ELT on Snowflake Data Cloud with Snowpark

PIPELINE EXAMPLE #1:

Ingest to Data Cloud Platform

Pipeline Overview

Data Cloud platforms are a critical component of modern data architecture in enterprises that leverage massive amounts of data to drive the quality of their products and services. A foundational practice to receive value from your Data Cloud is to ensure that it is filled with current and reliable data. This often involves batch ingestion of large amounts of data to feed your Data Cloud environment initially. This is often performed by migrating large amounts of raw data into an object store like Amazon S3, Azure ADLS, or Google Cloud Storage. This way, data can be transformed through processing operations to be in the best format for reliably filling your data cloud.

Key Steps

- Read web logs stored on Amazon S3
- Convert data types of certain fields from string to their appropriate types
- Enrich records by creating new fields using regular expressions
- Store the transformed web logs in Snowflake Data Cloud

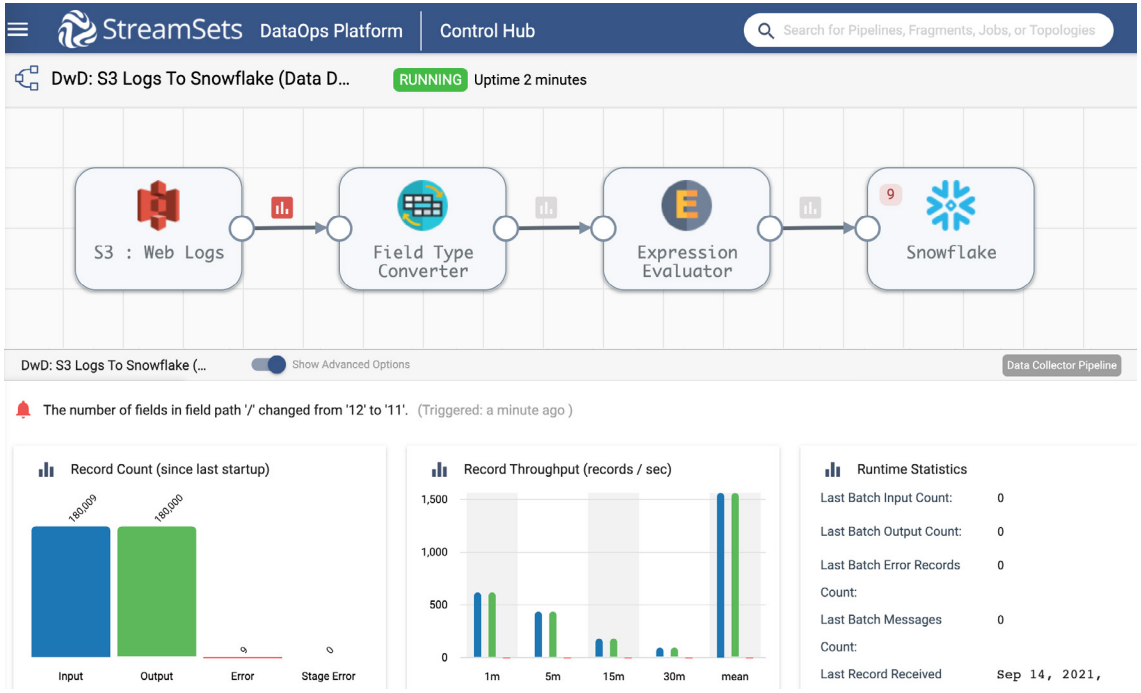

Figure 3. Pipeline Example #1 – Ingest to Data Cloud Platform

Smart Data Pipelines at Work

Built-in Log Parser

- The directory origin has built-in capability of parsing logs in various formats such as Common Log Format, Apache Error Log Format, Combined Log Format, Apache Access Log Custom Format, Grok Pattern-based Format, etc.

Data Enrichment

- Data types can be converted from one format to a different format without having to write custom code/script. For example, string to date formats with and without time zones, integers, floating point numbers, etc.
- New fields can be created based on existing fields and/or certain conditions to enrich the records using expressions.

Schema Evolution

- Snowflake destination can be configured to auto-create new columns or tables if they don't already exist.
- This eliminates the overhead of creating the tables beforehand especially when the source schema is unknown.

Real-time Transformations

- As data flows through the pipeline, it is transformed in real-time to enable downstream consumption and/or to meet business requirements. This process works independently of the source, destination, data formats, and processing modes -- streaming, batch, or micro-batch.

Offset Tracking

- Internal offset tracking imposes a "state" on the pipeline which effectively allows the pipeline to be stopped and restarted in a "pick up where you left off" manner.
- External offset tracking enables failover via the control plane (StreamSets Control Hub) at the execution engine / data plane (StreamSets Data Collector) level.

Dataflow Metrics

- The platform captures data flow metrics in a time series manner to enable real-time insights at the pipeline and individual stage level. Some of the metrics exposed in the UI include input and output record counts, record throughput, stage level processing times etc.
- This also enables configuring data SLAs.

Preview and Snapshots

- The platform has a built-in ability to preview data before running the pipeline and being able to step through each transformation (stage by stage) against the data being read from the source.
- The platform has a built-in ability to take snapshots of data in real time while the pipeline is running, without constraining throughput and performance. This enables debugging and replaying of dataflows to identify issues, in data transformations, for example, in production environments.

Handling Semantic Drift

Now let's assume that the structure of the log file changes. For example, the order of the columns change as new files are uploaded or added for processing. In that case, the pipeline would continue to work without rewriting any of the pipeline logic. In other words, the data enrichment stages (i.e., Field Type Converter and Expression Evaluator) would continue to transform and enrich the data without making any changes.

PIPELINE EXAMPLE #2:

Change Data Capture from Legacy to Data Cloud Platform

Pipeline Overview

Syncing incremental data is the next critical step in migrating to the Data Cloud platform. After you load your initial data sets it's important to keep data fresh and updated through ongoing change data capture operations. In StreamSets, this is as easy as leveraging one of our prebuilt CDC origins. By leveraging smart clients to listen for new and changing data actively, you can ensure that data from your core systems is continuously replicated into your Data Cloud platform. StreamSets provides out-of-the-box CDC enabled sources to easily develop and automate Change Data Capture (CDC) operations.

Key Steps

- Configure Oracle CDC client
- Select stream selector processor to route records
- Mask for PII data that comes into stream
- Configure Snowflake destination

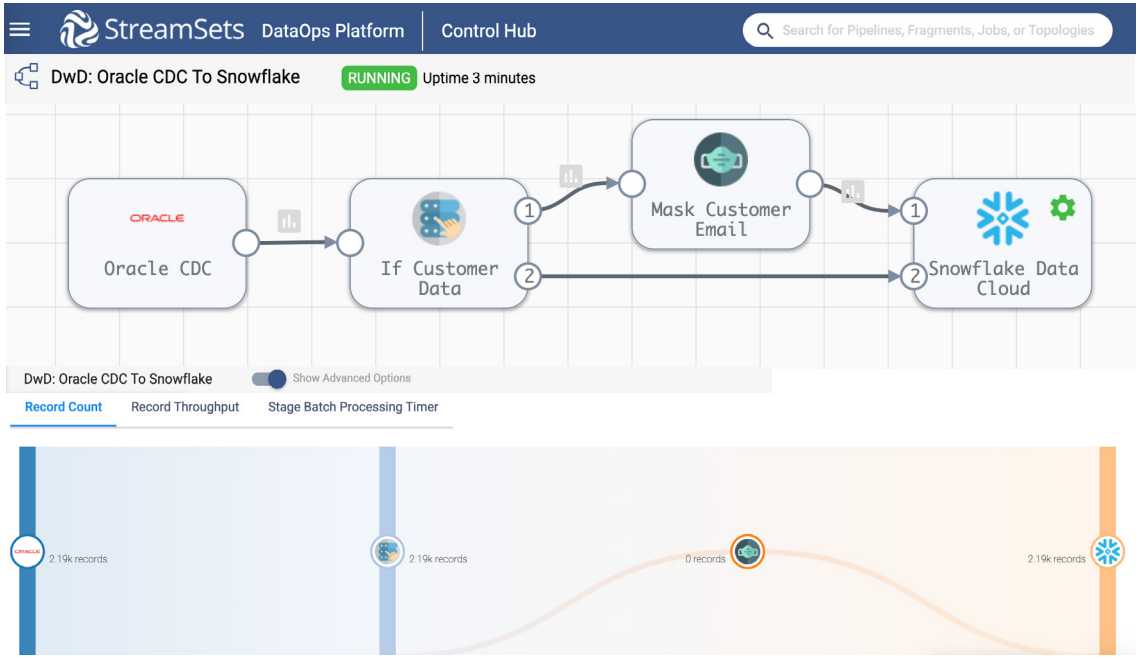

Figure 4. Pipeline Example #2 – Change Data Capture from Legacy to Data Cloud Platform

Smart Data Pipelines at Work

Select Authoring Data Collector

- Once the pipeline has been imported, open it in the pipeline canvas, and select Authoring Data Collector -- this is the Data Collector engine that would have been deployed once your deployment was successfully activated.

Oracle CDC Client Origin

- Oracle CDC Client origin will enable you to capture Create, Update, and Delete (CRUD) operations across various tables in your Oracle data warehouse so that your Snowflake Data Cloud can be kept in sync.

Stream Selector Processor

- This processor will conditionally route records based on user-defined conditions. For instance, in this case, we'd like to protect customer email addresses from being ingested (in plaintext) in Snowflake.

Field Masker Processor

- This processor will enable us to "mask" PII in configured fields. In this case, it is configured to mask customer email addresses before sending it over to Snowflake.

CDC to Snowflake Destination

- Snowflake destination uses the MERGE command to write CDC data that's being captured from Oracle. Note that this provides real-time access to data as it is written into Snowflake tables.

Pipeline Validation, Preview and Test Run

- Once you've updated the pipeline parameters, you can validate it to make sure the credentials are correct, preview the data to make sure the transformations are accurate, and test run the pipeline to ensure the data is being ingested into Snowflake correctly.

Create and Run Job

- Jobs enable you to execute, manage and orchestrate data pipelines that run across multiple engines. You can increase the number of pipeline instances that run for a job or enable a job for pipeline failover to minimize downtime due to unexpected failures.

Check Formatting

- For configuration details such as Staging, **Snowflake File Format**, defaults for missing fields, etc. refer to the configuration section." under **"CDC to Snowflake Destination"** key step.

Automatic Change Data Capture Sources

CDC-enabled origins can read change capture data. Some exclusively read change capture data; others can be configured to read it. When reading changed data, they determine operations associated with the data - such as insert, update, upsert, or delete. Using a CDC-enabled origin in a pipeline allows you to write changed data from one system to another easily. You can also use a CDC-enabled origin to write to non-CRUD destinations and non-CDC origins to write to CRUD-enabled stages. By providing pre-built CDC origins, users can easily and reliably build change data capture solutions without the heavy tax of hand coding the origins or bolting on heavy enterprise solutions that are not built for modern cloud environments.

Oracle CDC to Snowflake Sample Pipeline

After you download the sample pipeline from GitHub, use the Import a pipeline feature to create an instance of the pipeline in your StreamSets DataOps Platform account. For a full step-by-step walkthrough visit the Oracle CDC guide.

PIPELINE EXAMPLE #3:

Streaming Files Into Snowflake Data Cloud Using Apache Kafka

Pipeline Overview

For companies to leverage continuous data delivery, it's important also to have solutions that address real-time streaming data. In many platforms, this means a separate tool, separate development interface, and a separate management and control console. StreamSets provides a single platform with a congruent developer experience fit for batch, streaming, ELT, and machine learning workloads. This means that data engineers don't have to spend hours toggling between different interfaces to develop a batch and streaming solution. Let's explore a common example of streaming using files as the origin.

Key Steps

- Configure Directory Source
- Remove and order fields
- Configure File Type Converter
- Configure Snowflake destination

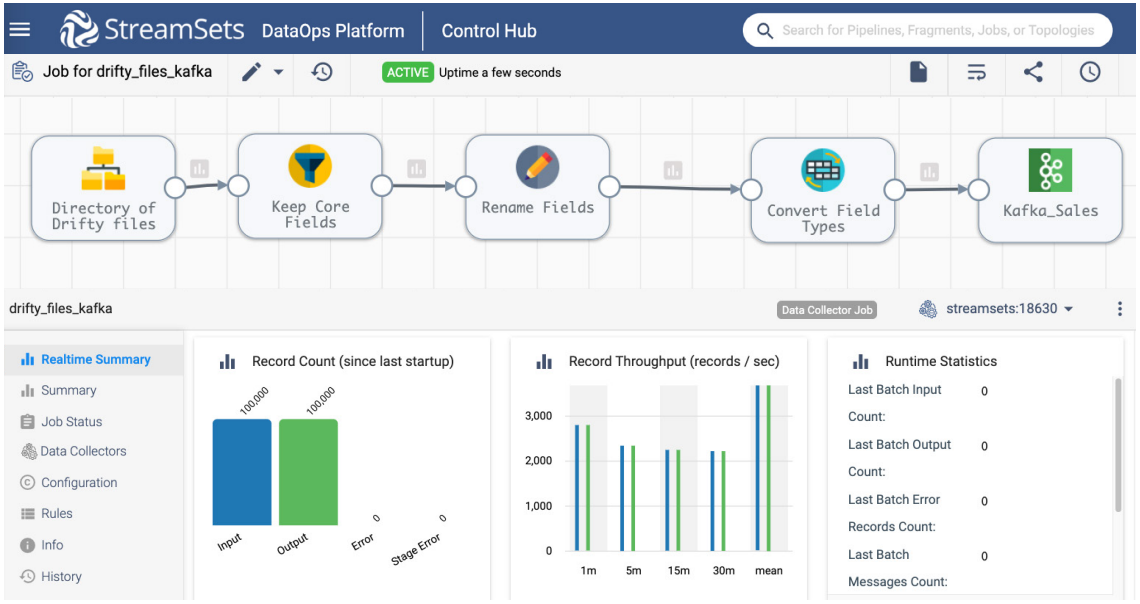

Figure 5. Pipeline Example #3 – Streaming Files Into Snowflake Data Cloud Using Apache Kafka

Smart Data Pipelines at Work

File Directory (Origin Settings)

- When beginning choose the Files Directory source and specify the path to the root directory where your files will be stored.

Select Data Format

- Select Delimiter Format Type: Default CSV Parser: Apache Commons.

FieldRemover Processor: Keep Core Fields

- In this pipeline, you are telling it what fields you want to process; all others will be ignored. With this configuration, if your source files come in with additional columns that may or may not be consistent between file generation, it doesn't matter. Those added fields won't make their way into your target nor into other downstream processors.

FieldOrder Processor Settings: Order for Table

- The Field Order processor allows you to tell the pipeline the order you want the fields to be processed downstream, in our case to Snowflake. The fields can either be manually entered or picked from the list once you run the Preview. By default, StreamSets will Send record to error. Another option is to add a Default Value. Or, you can choose to Discard.

FieldTypeConverter Processor

- The Field Type Converter allows you to convert a single or list of fields to a particular Data Type. You can keep adding more conversions until you have all the fields set to the Data Types and formats you want.

Configure Snowflake Destination

- We are now ready to load our data into Snowflake.

Select Data Drift Detection

- With Data Drift detection enabled, new data fields can simply show up and the downstream team can decide if they want to utilize them or not. So, as new fields are added, you've just eliminated the need to update your job, go back through QA / Stage / Perf environments, and then finally re-release into production.

Select Table Auto Create

- If checked, a Table will be created in the target environment if it does not already exist.

Multi-table Inserts

StreamSets DataOps Platform can handle Semantic and Structural Drift. In other words, even if the field order changes from file to file, it won't affect the logic or flow of your pipeline to your destination. To handle this change on the Snowflake side, StreamSets has included the capability for Multi-table Inserts. These inserts provide two major advantages. First, when initially loading your data into Snowflake, you don't need to engineer for the schema of the source system. Simply design the pipeline and hit play, and the tables will auto-create based on the source schema. This can save you weeks to months in migrating your data to your Data Cloud. Secondly, multi-table inserts help when handling data drift. As data structure changes over time it doesn't take down your smart data pipeline. The new row or column is simply populated in the destination table.

PIPELINE EXAMPLE #4:

Native ELT On Snowflake Data Cloud With Snowpark

Pipeline Overview

We have explored how to land raw data into your Data Cloud platform. But as you know, many times analysts, data scientists, and other key peers need their data in a more conformed structure than is provided with raw data. Data engineers working in the Data Cloud ecosystem aim to perform these critical transformations with as little movement and overhead as possible. This is because every time data is moved or transferred, we open ourselves up to the chance for data to drift. The StreamSets engine **Transformer for Snowpark** is built to implement complex end-to-end ELT workloads directly on the Snowflake Data Cloud without moving data outside of the Data Cloud. As an example, data engineers can use the engine to denormalize and aggregate data across several tables in Snowflake. In the following example, the data pipeline is designed to join across master-detail tables.

Key Steps

- Configure Snowflake Source (RAW Data)
- Join multiple tables
- De-normalize records
- De-duplicate columns
- Aggregate and store results

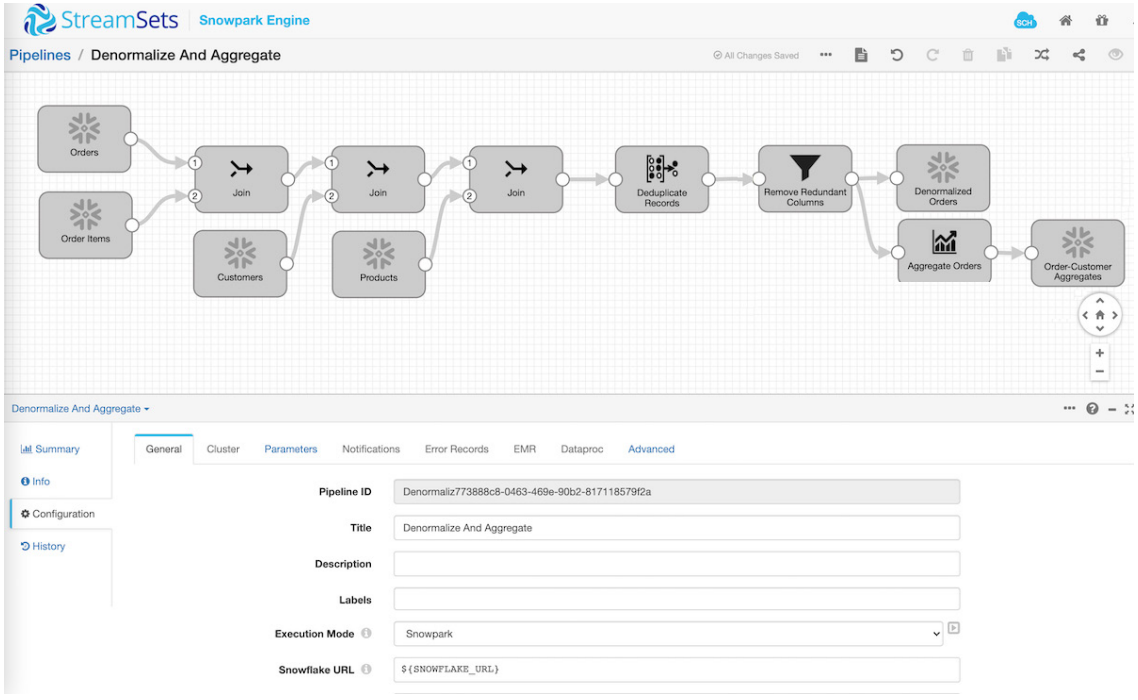

Figure 6. Pipeline Example #4 – Native ELT On Snowflake Data Cloud With Snowpark

Smart Data Pipelines at Work

Snowflake Table (Origin Settings)

- When beginning, choose the Snowflake source and specify the path to the target tables you aim to access.

Setup Separate Join Operations

- Joins include Orders and Order Items, respectively, and also with foreign key constraints Products and Customers using multiple Join processors. To note, If the destination tables don't already exist on Snowflake, they are created automatically during pipeline execution.

FieldRemover Processor for Redundant Columns

- Because the resulting denormalized records would include redundant columns like CUSTOMER_ID from Orders table and Customers table. For example, a Field Remover processor is injected into the pipeline to remove such columns.

Configure Snowflake Destination to Store Denormalized Records

- Select Table Auto Create. If checked, a Table will be created in the target environment if it does not already exist.

Aggregator Processor

- Perform aggregations to calculate the total order amount per order per customer from information that's coming in from three different tables — ORDERS, ORDER_ITEMS, and CUSTOMERS.

Configure Snowflake Destination to Store Aggregates

- Select Table Auto Create. If checked, a Table will be created in the target environment if it does not already exist.

CI/CD and Testing for Your Data Pipelines

Data transformations are a vital part of delivering continuous data. However, how do you know that the pipeline and ETL operations you built are performing as expected? The StreamSets SDK for Python and StreamSets Test Framework give you dynamic tools for testing how your pipeline works even as it is actively developing. These frameworks can enable continuous development by giving data engineers assurance their pipelines are working as intended. You can dynamically create, execute, and monitor data pipelines directly from the SDK through coding like Python and also in the user interface.

Operationalizing Smart Data Pipelines

In a modern enterprise, pipeline development is only part of the battle. As your technology stack evolves, you will need to design pipelines for change and deploy them, monitor them continually, and refactor in an agile fashion. When managing thousands of data pipelines, getting visibility into all the pipelines and the performance across all stages can be a staggering proposition. Smart data pipelines give you continuous visibility at every stage of execution. Collections of pipelines can be visualized in live data maps and drilled into when problems arise. This drastically reduces the time data engineers spend fixing errors and hunting for root causes. Smart data pipelines let you make changes to pipelines, even when they are running in production, allowing you to create agile development sprints.

Smart data pipelines report on critical metrics including:

- Throughput rates
- Error rates
- Execution time by stage
- PII detection
- Schema drift alerting
- Semantic drift alerting

This active monitoring helps data engineers ensure that data is delivered correctly with retained fidelity. It also helps flag and troubleshoot any operational or performance issues with either the data pipelines

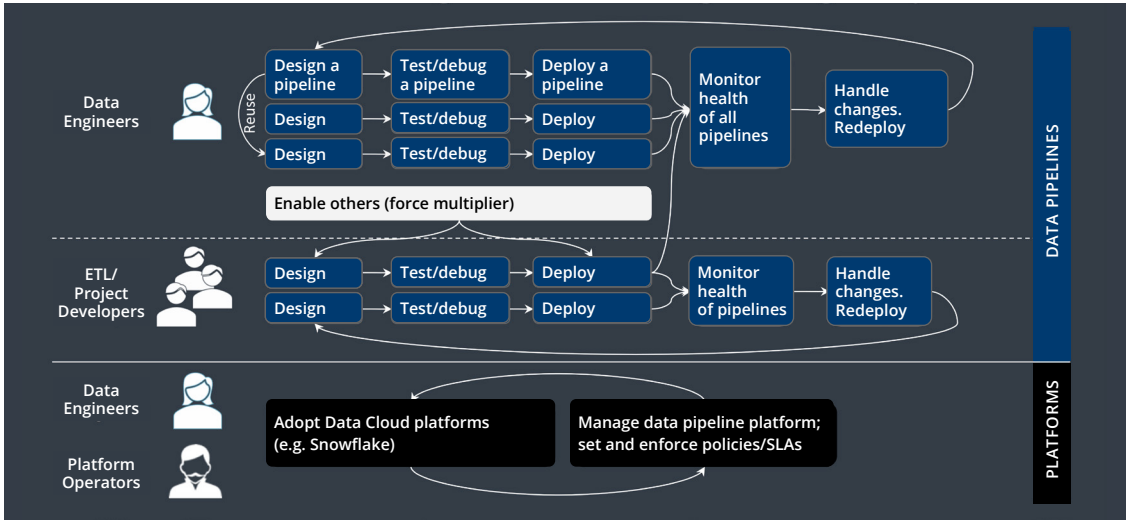

Figure 7. The full lifecycle of data pipelines: Design, Deploy, and Enrich

or the underlying execution engines in real time, no matter where they are deployed, even across multiple platforms both on-premises and in the cloud. Such end-to-end transparency significantly reduces the administrative burden of monitoring and managing tens of thousands of pipelines across hundreds of engines.

Real-time instrumentation is also critical for smart pipelines' operational resiliency to data drift. When drift happens, data engineers a) can detect it immediately, based on the sensors embedded into the smart data pipelines themselves, and b) have choices on how they want to handle the drift. In some cases, structural drift is not material to the meaning of the data, so the smart data pipeline can simply keep running with no change or intervention whatsoever. Other types of change, such as a schema update, can be automatically propagated into downstream systems. This ability to automatically handle many common types of data drift drastically reduces the time and effort spent on maintenance and change management of data pipelines in operation.

Other times, drift may be a material or even dangerous change, and the data may need to be diverted and reviewed by a data engineer or analyst. Smart pipelines can detect such changes and alert the relevant team member when they arise.

StreamSets: Smart Data Pipelines for Data Engineers

The StreamSets DataOps Platform supports your entire data team with an easy on-ramp for a wide variety of developers, and powerful tools for advanced data engineers. Our smart data pipelines are resilient to changes. The platform actively detects and alerts users when data drift occurs. StreamSets lets you change when business needs change and makes it easy to port data pipelines across clouds and data platforms without re-writes.

The platform consists of two powerful data engines and a comprehensive management hub:

StreamSets Control Hub is a single hub for designing, deploying, monitoring, managing, and optimizing all your data pipelines and data processing jobs. As the central nervous system of the StreamSets DataOps Platform, Control Hub:

- Lets your entire extended team collaborate, design, monitor, and optimize data pipelines and jobs running on Data Collector Engine and Transformer Engine
- Provides a real-time view of all the data pipelines across your enterprise
- Manages, monitors, and scales the Data Collector and Transformer engines themselves to optimize your overall StreamSets environment.
- Gives you complete transparency and control of all data pipelines and execution engines across your entire hybrid/multi-cloud architecture, in one single hub.

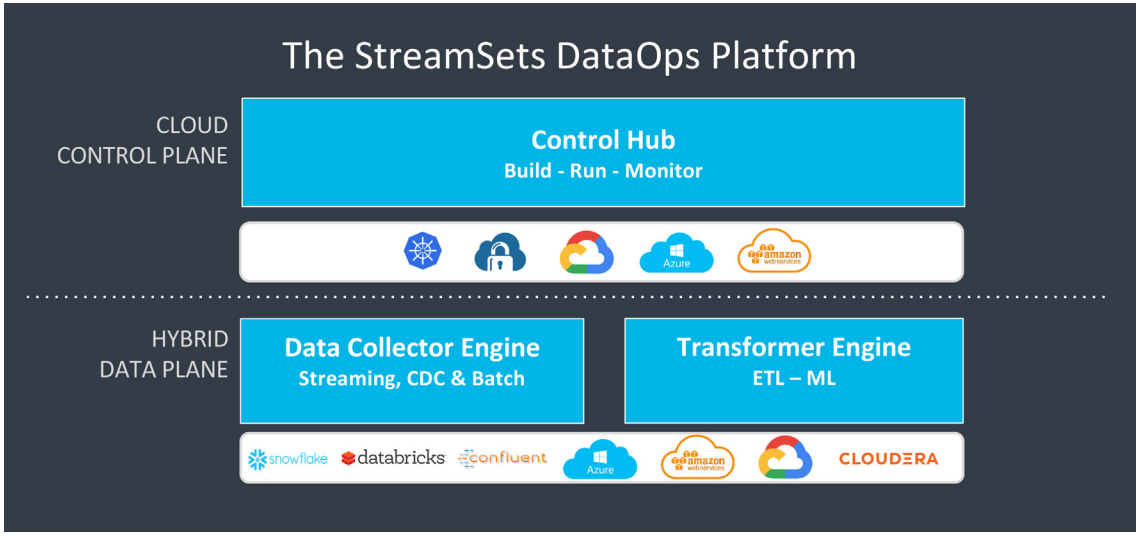

Figure 8. Decoupled Design: Control Plane and Data Plane

StreamSets Data Collector Engine is an easy-to-use data pipeline engine for streaming, CDC and batch ingest from any source to any destination. It lets your data engineers:

- Spend their time building data pipelines, enabling self-service, and innovating
- Minimize the time they spend maintaining, rewriting, and fixing pipelines.

StreamSets Transformer Engine is a data pipeline engine designed for any developer or data engineer to build ETL and ML pipelines that execute on **Snowflake via Snowpark** or on Apache Spark clusters and services. As an intent-driven visual design tool, it lets users more easily create pipelines for performing ETL and machine learning operations.

Conclusion

For data engineers, the Data Cloud provides numerous advantages to receive better data access to data, governance for your data, and actionable functions on your data. However, simply migrating your legacy data platform and legacy data pipelines to the Data Cloud brings all your problems along with it. Data pipelines for the Data Cloud need to address the elastic, scalable, and accessible nature of the Data Cloud. Smart data pipelines take full advantage of these Data Cloud attributes, while also detecting and being resilient to data drift.

By developing the core capabilities to land raw data into snowflake landing zones, incrementally load data from traditional sources, enrich with real-time data from streaming services and event hubs, and transform data to be delivered to analytics teams and platforms, you will have the foundations for delivering fast, reliable insight to every corner of your business. StreamSets helps you build smart data pipelines for the Data Cloud with a common design interface, extensive tools for deep integration, reliable operation with monitoring and reporting, and truly portable pipeline design across all environments.

Do you want to start building these design patterns today? Try StreamSets

About StreamSets

At StreamSets, our mission is to make data engineering teams wildly successful. The StreamSets DataOps Platform empowers engineers to build and run the smart data pipelines needed to power DataOps across hybrid and multi-cloud architectures. That's why the largest companies in the world trust StreamSets to power millions of data pipelines for modern analytics, AI/ML and smart applications. With StreamSets, data engineers spend less time fixing and more time doing.
To learn more, visit **www.streamsets.com** and follow us on **LinkedIn**.

START YOUR FREE TRIAL

StreamSets