



WHITE PAPER

Data Engineer's Handbook

4 Cloud Design Patterns for Data Ingestion and Transformation

Table of Contents

Introduction.	2
Critical Design Patterns for the Cloud.	2
The Role of the Data Engineer	3
The Rise of Smart Data Pipelines	4
4 Cloud Design Patterns	5
Pipeline Example #1: Ingest to Cloud Data Lakes or Cloud Storage	5
Pipeline Example #2: Ingest to Cloud Data Warehouses.	8
Pipeline Example #3: Ingest to Cloud Messaging Services or Event Hubs	11
Pipeline Example #4: Transform from Raw Data to Conformed Data.	14
Operationalizing Smart Data Pipelines	16
Conclusion.	20

StreamSets and the StreamSets logo are the registered trademarks of StreamSets, Inc.
All other marks are the property of their respective owners.

Introduction

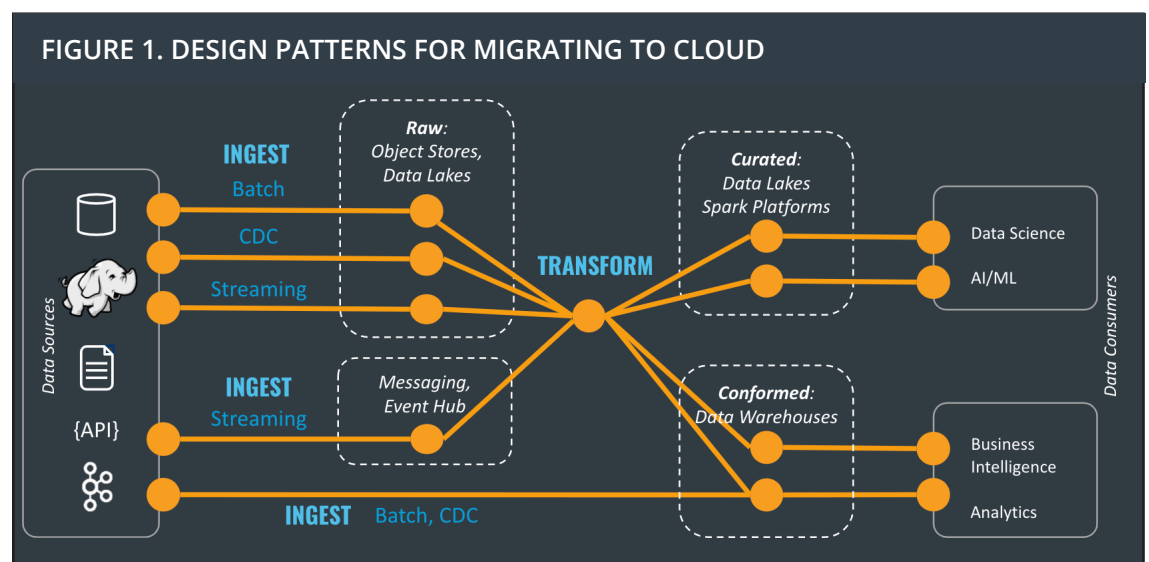
The move to the cloud has become top of mind and more urgent for data engineers than ever before. According to Gartner, “By 2023, 75% of all databases will be on a cloud platform.”¹ This mass migration to the cloud has translated into huge new cloud projects that have dropped into the lap of the data engineer.

Whether you are an individual contributor or a senior data team leader, you most likely are supporting a growing number of ETL developers who in turn support an expanding number of data scientists and analysts across the whole business. They want data faster, and they increasingly want data from outside the company. The cloud is the easiest way to gather all of the external data your teams need and let the analytics team free to party on the data. For data engineers moving to the cloud means pipeline redesigns, migration projects, and shifts in data processing strategy.

Critical Design Patterns for the Cloud

To successfully migrate data and data workloads to your cloud data platforms these are the four most common data pipeline design patterns:

- Ingesting to Cloud Data Lakes and Cloud Storage
- Ingesting to Cloud Data Warehouses
- Ingesting to Cloud Messaging Services or Event Hubs
- Transforming from Raw Data to Conformed Data



¹ <https://www.gartner.com/en/newsroom/press-releases/2019-07-01-gartner-says-the-future-of-the-database-market-is-the>

These four data pipeline patterns are the building blocks for ingesting, transforming and migrating your data into cloud data platforms. Together, they help data engineers accelerate and simplify the move to the cloud in support of next generation data analytics, data science, AI, and machine learning workloads.

This handbook will walk you through the 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. From batch ingestion, to change data capture, to real-time streaming, all workloads can be managed and optimized through interactive maps.

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 are able to 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 that is a result of modern data movement. Keeping up with data drift creates a huge burden on data engineers and platform operators, both 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 public cloud services, 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.

“Most people don’t realize that change and evolution of target systems causes 10-20 hours of work for the data engineer. Every single change.”

The Rise of Smart Data Pipelines

A smart data pipeline is a data pipeline that is designed to operate in a continuous fashion, with as little manual intervention as possible. Smart data pipelines are essential in highly dynamic cloud data environments where data flows across 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 cloud, we look at the difference smart data pipelines make and how they adapt to change.

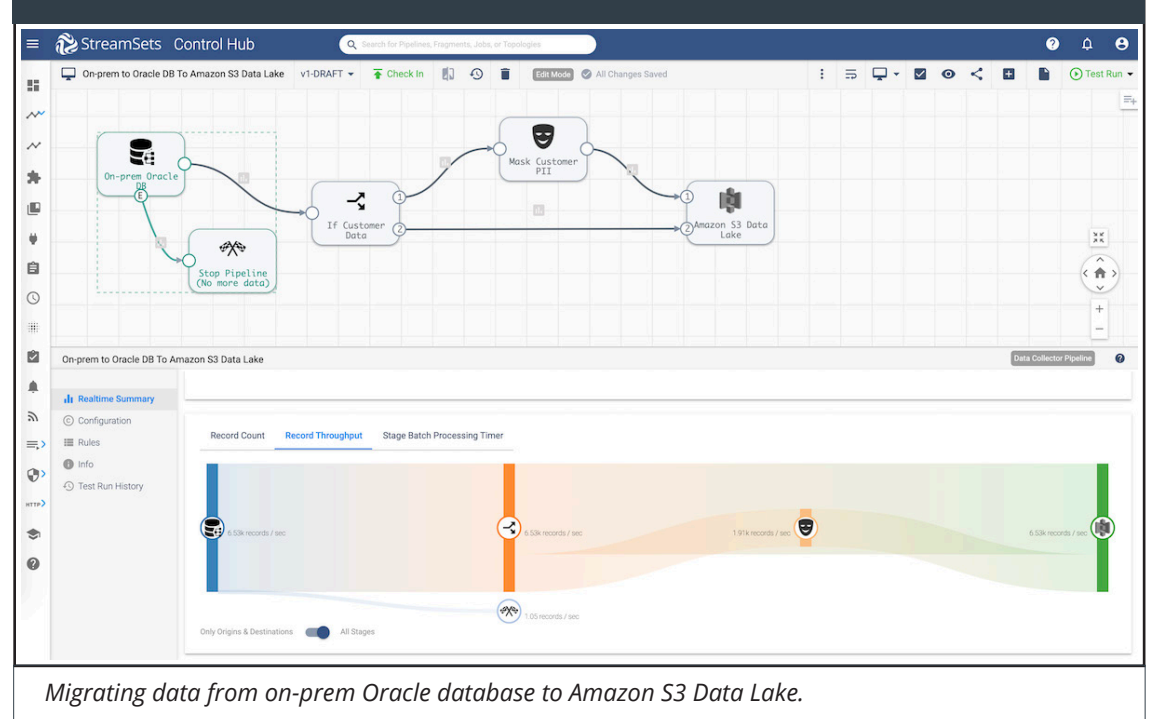
4 Cloud Design Patterns

Pipeline Example #1: Ingest to Cloud Data Lakes or Cloud Storage

Pipeline Overview

The first pattern is the most common and often the first step in moving data to the cloud. It's all about ingesting data into cloud data lake or other raw cloud storage. This is the gateway into the cloud for much of your data — it can go in many different directions and for many different use cases after it's ingested into the cloud data lake or cloud storage.

FIGURE 2. PIPELINE EXAMPLE #1 – INGEST TO CLOUD DATA LAKES



Key Steps

- Read data from multiple tables in parallel from the Oracle database.
- Conditionally route customer records based on the table name record header attribute (metadata) exposed by the platform to mask customer PII. For example, customers' email addresses.
- Securely store the data on Amazon S3 Data Lake using server-side encryption and partitioned by table name.
- Automatically stop the pipeline once all of the data has been processed from Oracle database and written to Amazon S3.

Smart Data Pipelines at Work

- 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.
- Multiplexing and Demultiplexing
 - Same pipeline is configured to read multiple tables with different schemas from the same database, and then seamlessly writes records to the appropriate partitions. Optionally, it can be configured to include or exclude certain schemas and tables based on name patterns using the SQL LIKE syntax.
- 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 as well as 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’s 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’s built-in 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 Infrastructure Drift

Now let's assume you need to ingest data from MySQL database instead of Oracle database, while all other requirements remain the same. In this particular scenario, here are the options that would require minimal pipeline development time and effort:

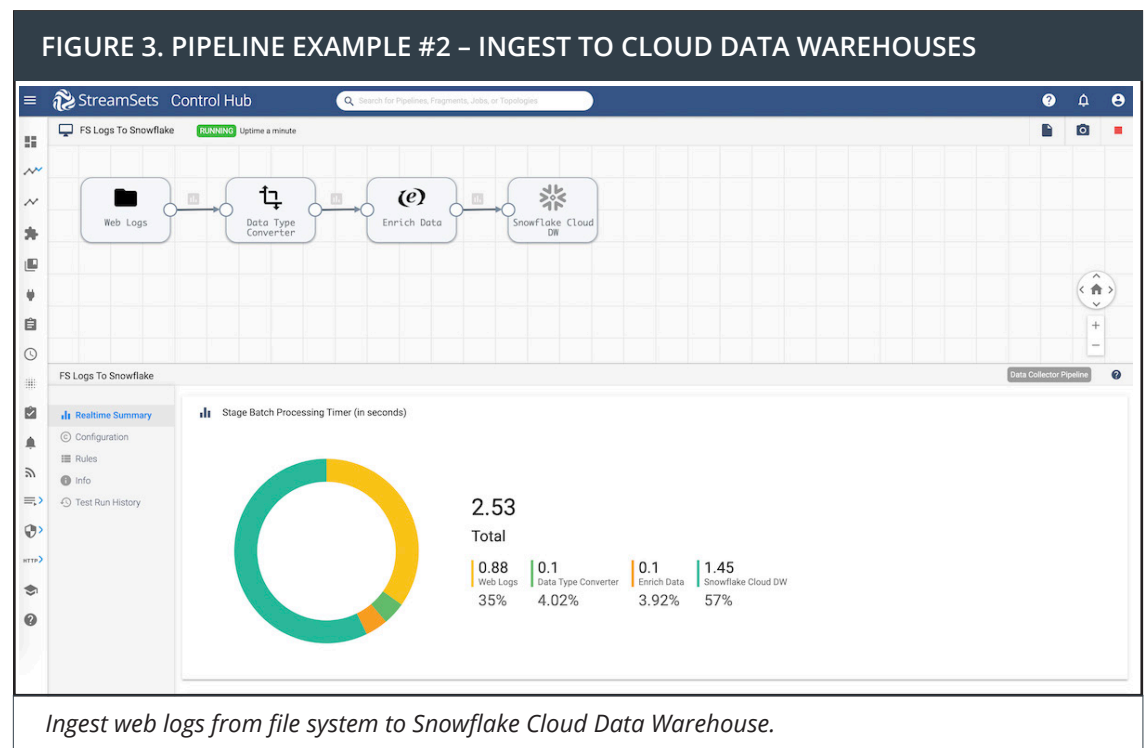
- Duplicate pipeline:
 - You can easily duplicate the pipeline and then update the origin to ingest data from MySQL instead of Oracle. The key attributes to change would be the JDBC URL and credentials.
 - This operation will create a new instance of the pipeline so you can also make other changes, if needed, and still have the original pipeline "as-is".
- Create a new version of the pipeline:
 - Instead of creating a new pipeline (by way of duplication), you could also easily create a new version of the same pipeline.
 - The different versions of the pipeline are tracked and maintained by the built-in version control system.
 - This operation will not create a new instance of the pipeline, but you can still revert back to the older version(s) or visually compare versions in the UI.
- Parameterize key attributes:
 - Instead of duplicating the pipeline or creating a new version, you could also parameterize key attributes, such as JDBC URL and credentials. Then create a job that will run multiple instances of the same pipeline by passing in different parameters to connect to Oracle or MySQL database.

"If you're new to engineering or Hadoop, that comes with experience and time. No junior engineer will build all the fault safety things."

Pipeline Example #2: Ingest to Cloud Data Warehouses

Pipeline Overview

Cloud data warehouses are a critical component of modern data architecture in enterprises that leverage massive amounts of data to drive quality of their products and services. They are a new breed that bring the added advantage of cost-effectiveness and scalability with pay-as-you-go pricing models, a serverless approach, and on-demand resources made possible by separating compute and storage to provide a layer specifically for fast analytics, reporting and data mining.



Key Steps

- Read web logs stored in a file system.
- 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 Cloud Data Warehouse.

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 various formats 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 as well as 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's 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's 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 having to rewrite any of the pipeline logic. In other words, the data enrichment stages (for example, Field Type Converter and Expression Evaluator) would continue to transform and enrich the data without making any changes.

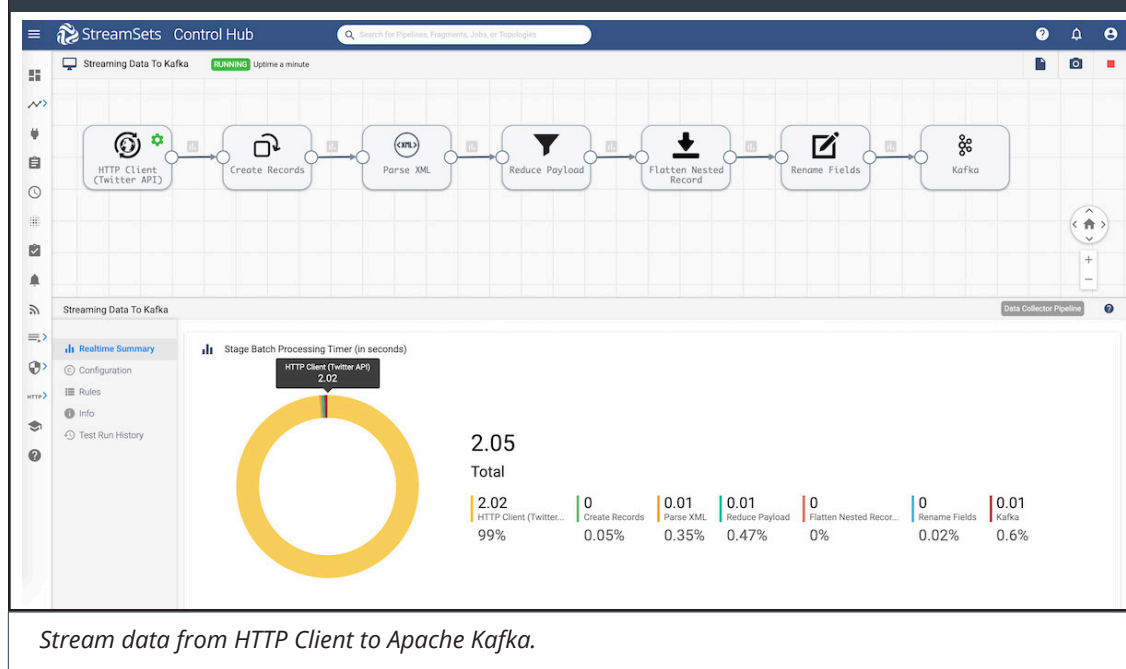
“The major slowdown is moving from all on prem, moving that to full cloud is a rude awakening, It's not what you think. You can't just lift and shift some things.”

Pipeline Example #3: Ingest to Cloud Messaging Services or Event Hubs

Pipeline Overview

Ingesting into the cloud messaging systems enables flexible deployment and maintenance of decoupled producers and consumers. This design pattern eliminates duplication of logic and, in fact, allows for decoupled logic and operations for different consumers. In the case of Kafka, for example, topics provide a well understood metaphor, partitions provide a basis for scaling topics, and consumer groups provide “Shared Consumer” capabilities across threads, processes or both.

FIGURE 4. PIPELINE EXAMPLE #3 – INGEST TO CLOUD MESSAGING SERVICES OR EVENT HUBS



Key Steps

- Stream data from Twitter using Twitter API.
- Create individual tweet records from the list.
- Enrich records by parsing and extracting data from XML field.
- Reduce payload by removing unwanted fields.
- Flatten nested records.
- Send transformed data to Kafka.

Smart Data Pipelines at Work

- Built-in Field Pivoter and XML Parser
 - When external systems return data in nested or other formats, the records can be 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.
- Dataflow Metrics
 - The platform captures data flow metrics in a time series manner to enable real-time insights at the pipeline as well as 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's 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's 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 Infrastructure Drift

Now let's assume that you need to test a new version of Apache Kafka while continuing to support older versions for the following reasons:

- Existing systems that depend on older versions continue to run.
- Company IT policy or business requirements.
- It continues to serve other systems that depend on older versions.

In such cases, you have the following options that require minimal development efforts:

- Create a new version of the pipeline and select the new Kafka version in the destination in the new pipeline. The different versions of the pipeline are tracked and maintained by the built-in version control system. This operation will not create a new instance of the pipeline, but you can still revert back to the older version(s) or visually compare versions in the UI.
- Duplicate the pipeline and select the new Kafka version in the destination in the duplicated pipeline. This operation will create a new instance of the pipeline so you can also make other changes, if needed, and still have the original pipeline "as-is".
- In the same pipeline, add a second Kafka destination and select the new Kafka version for it.

Either of these options can be implemented within minutes without having to rewrite any of the pipeline logic.

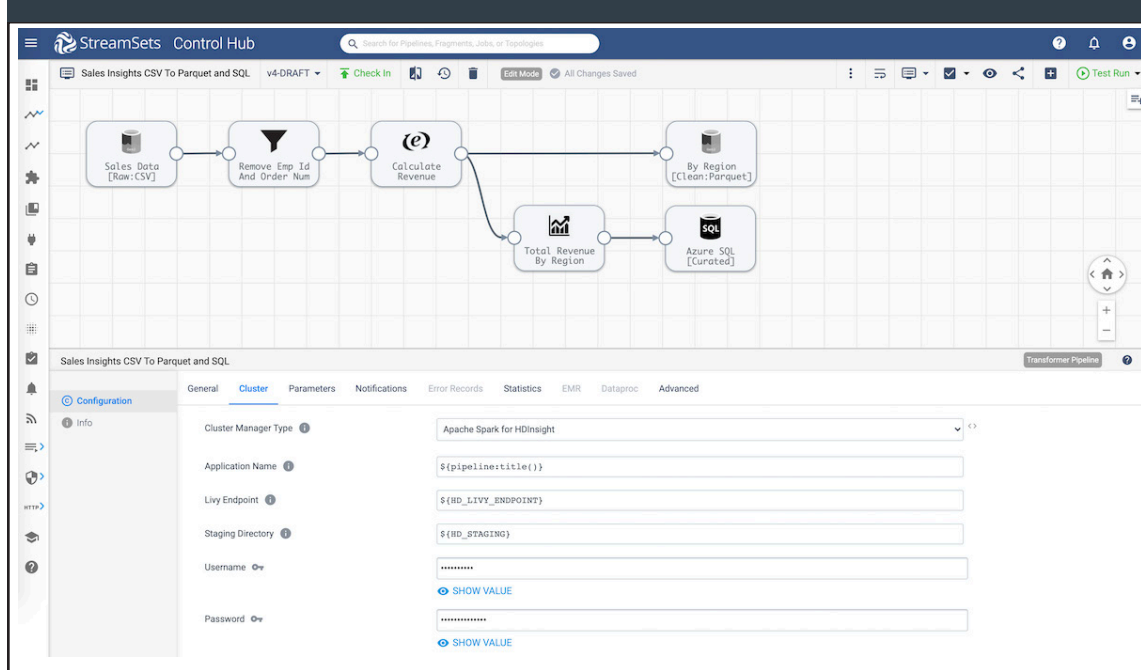
"I have some on prem, some on cloud, some on SaaS. How do I put it all together to build value out of all this data in a cost-effective way? The cost element comes in afterwards — you don't think about it up front."

Pipeline Example #4: Transform from Raw Data to Conformed Data

Pipeline Overview

Raw zone normally stores large amounts of data in its originating state usually in its original format, such as JSON or CSV. Clean zone can be thought of as a filter zone that improves data quality and may involve data enrichment. Common transformations include data type definition and conversion, removing unnecessary columns, etc. to further improve the value of insights. The organization of this zone is normally dictated by business needs — for example, per region, date, department, etc. Curated zone is the consumption zone, optimized for analytics and not so much for data processing. This zone stores data in denormalized data marts and is best suited for analysts or data scientists that want to run ad hoc queries, analysis, or advanced analytics.

FIGURE 5. PIPELINE EXAMPLE #4 – TRANSFORM RAW DATA TO CONFORMED DATA



Ingest raw sales insights in CSV, perform transformations and aggregations and store the conformed (clean and curated) data in Parquet and SQL.

Key Steps

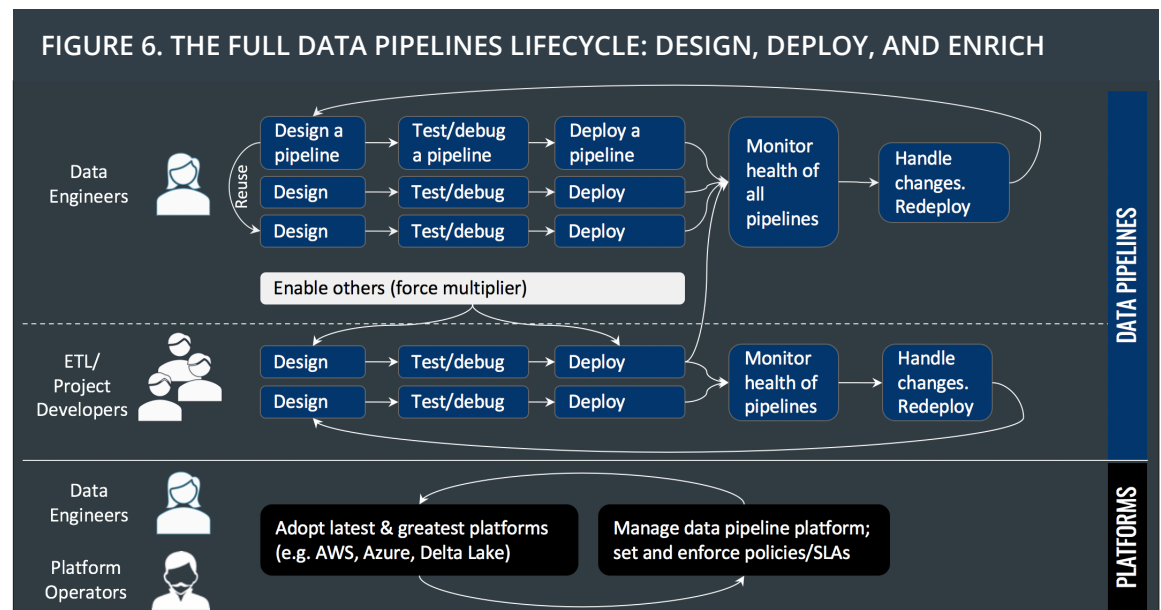
- Ingest Sales insights stored in CSV format on Azure Data Lake Storage (ADLS) Gen2.
- Remove information from records not critical for downstream analysis.
- Enrich records by performing calculations and aggregations.
- Store clean data in parquet format in ADLS Gen2 and aggregate data in Azure SQL.

Smart Data Pipelines at Work

- Multiple Formats
 - Same pipeline converts data into multiple formats and serves different type of downstream analysis.
- Multiple Cluster Manager Types
 - Transformer pipelines can be designed to run on various Apache Spark cluster types such as Azure HDInsigt, Amazon EMR, Google Dataproc, Databricks, Hadoop YARN, etc.
- Dataflow Metrics
 - The platform captures data flow metrics in a time series manner to enable real-time insights at the pipeline as well as 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
 - 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.
 - 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.

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 amount of 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
- Apache Spark errors
- 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 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.

Having this real-time instrumentation is also critical for smart pipelines' operational resiliency to data drift. When drift happens, data engineers a) are able to 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 amount of 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.

With multiple layers of operational resiliency built into the system, data engineers can feel confident that the data they are delivering to their analytics teams is sound and if any discrepancies happen they can trace the full execution back to the point of remediation.

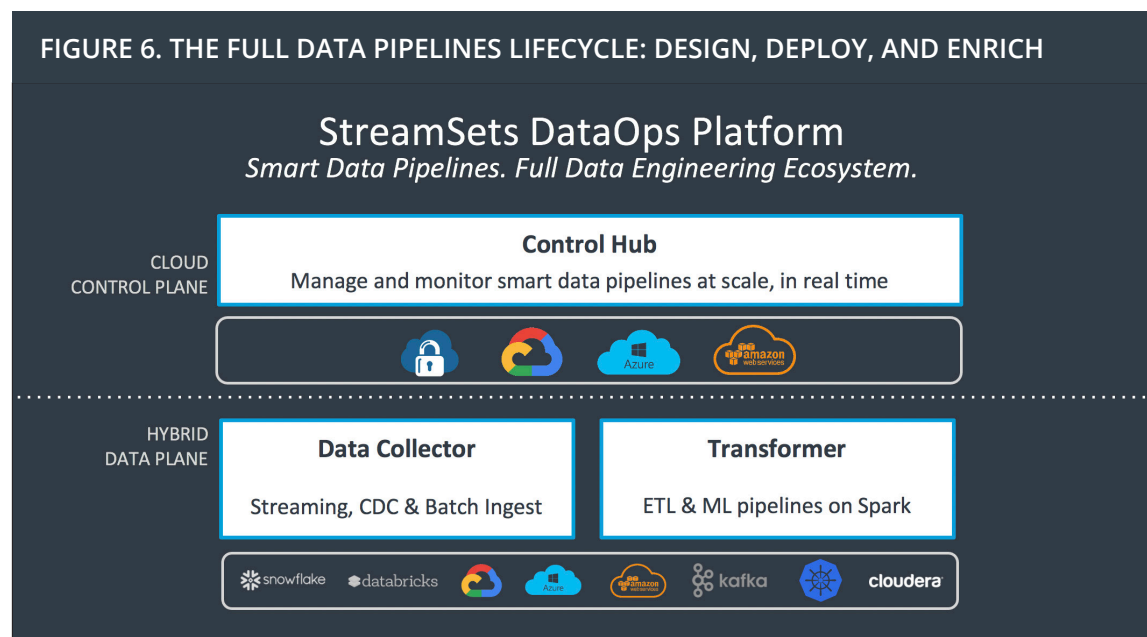
Critical Design Patterns for Modern Analytics

When deployed using smart data pipelines and a DataOps approach to operationalization, these critical design patterns help data engineers accelerate and simplify the move to the cloud in support of next generation data analytics, data science, AI, and machine learning workloads. StreamSets is the only data engineering platform for designing, deploying, and managing smart data pipelines across hybrid and cloud data architectures with ease.

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 the business needs change, and port data pipelines across clouds and data platforms without re-writes.

FIGURE 6. THE FULL DATA PIPELINES LIFECYCLE: DESIGN, DEPLOY, AND ENRICH



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

StreamSets Data Collector is an easy-to-use data pipeline engine for streaming, CDC, and batch ingest from any source, to any destination. Data engineers can spend their time building data pipelines, enabling self-service, and innovating, and minimize the time they spend maintaining, rewriting, and fixing pipelines.

StreamSets Transformer is a data pipeline engine designed for any developer or data engineer to build pipelines that execute on Apache Spark. Using an intent-driven visual design tool, users can create pipelines for performing ETL and machine learning operations. It allows everyone to harness the power of Apache Spark by eliminating the need for Scala and PySpark coding.

StreamSets Control Hub is a single hub for designing, deploying, monitoring, managing and optimizing all your data pipelines and data processing jobs. The central nervous system of the StreamSets DataOps Platform, Control Hub lets the entire extended team collaborate to design, monitor and optimize data pipelines and jobs running on Data Collector and Transformer, and gives you a real-time, end-to-end view of all data pipelines across your enterprise. Control Hub also manages, monitors, and scales the Data Collector and Transformer engines themselves to optimize your overall StreamSets environment. Control Hub provides full transparency and control of all data pipelines and execution engines across your entire hybrid/multi-cloud architecture in one single hub.

Conclusion

For data engineers, the public cloud provides numerous advantages to modernize your toolkit with exciting new data services that scale way beyond the confines of your traditional role. However, simply shifting your legacy platform to the public cloud brings all your problems along with it. Data pipelines for the cloud need to address the elastic, scalable, and accessible nature of the cloud. Smart data pipelines take full advantage of these cloud attributes, while also detecting and being resilient to data drift.

By developing the core capabilities to land data into raw data lakes and data warehouses, 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 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

StreamSets built the industry's first multi-cloud DataOps platform for modern data integration, helping enterprises to continuously flow big, streaming and traditional data to their data science and data analytics applications. The platform uniquely handles data drift, those frequent and unexpected changes to upstream data that break pipelines and damage data integrity. The StreamSets DataOps Platform allows for execution of any-to-any pipelines, ETL processing and machine learning with a cloud-native operations portal for the continuous automation and monitoring of complex multi-pipeline topologies. To learn more, go to: streamsets.com.

StreamSets and the StreamSets logo are the registered trademarks of StreamSets, Inc. All other marks reference are the property of their respective owners.