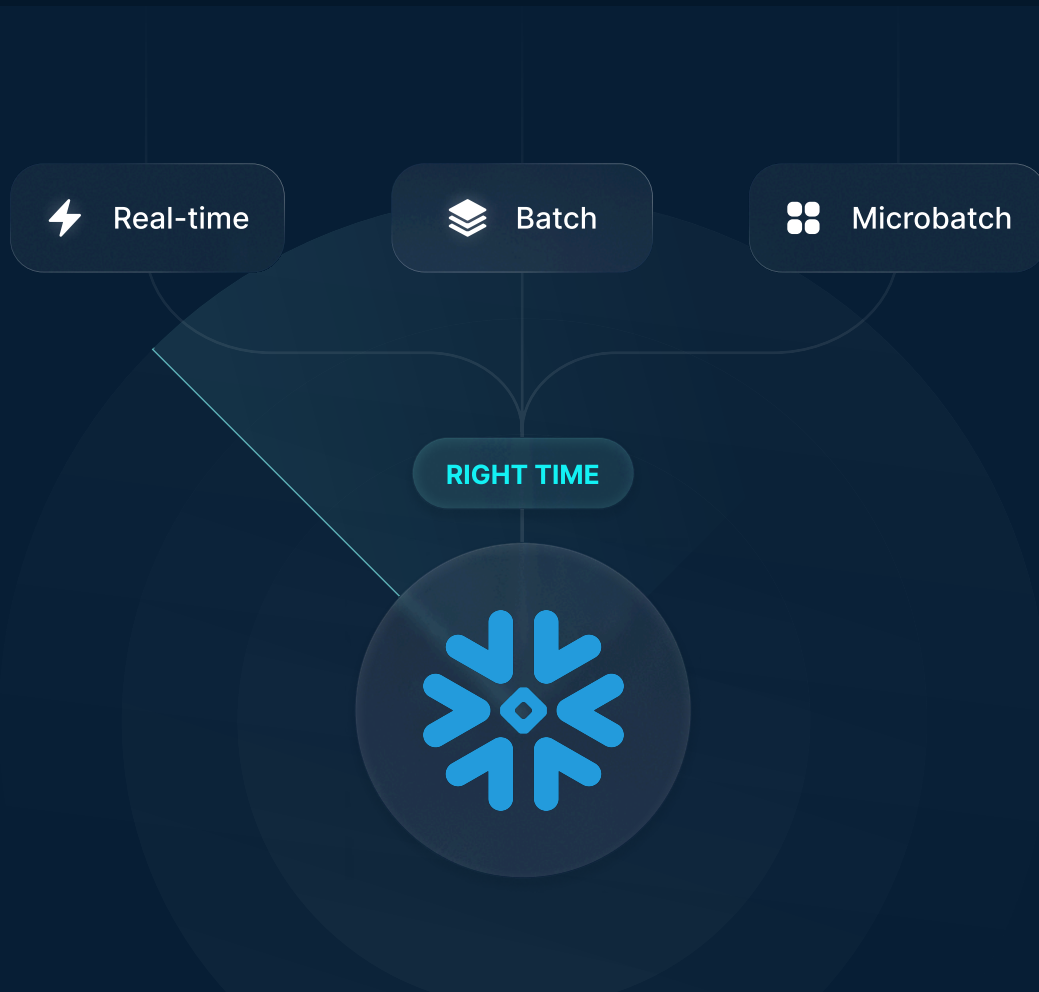





The Complete **Right-Time** Data Integration Guide for Snowflake



Introduction




If you're looking to integrate with Snowflake, there's no better time: ingestion options abound while Snowflake continues to make waves as one of the top cloud-agnostic data warehouse and analytics platforms.

However, with all those options come an equal number of choices. Choosing the right data ingestion option for Snowflake is about choosing:

-  The right timing for your data
-  The right price for your budget
-  The opportunities the integration opens up—or closes off

This guide offers a whirlwind tour of data integration options for Snowflake. We'll cover some of what's involved in several native and third-party integration options to give a taste of the integrations' required scope and features, as well as covering considerations on making the choice that's right for you.

You can read the guide all the way through or skip around to the sections that interest you most. Some suggestions for can't-miss topics by role include:

-  **Tech leaders:** Start by learning about **right-time data**, see where your organization falls on the right-time maturity scale, and find tips on choosing the right Snowflake solution
-  **Budget planners:** Consider the **total cost of ownership** for a Snowflake integration and how right-time data can lower costs
-  **Data engineers:** Jump directly into the **Snowflake data integration guides**, either native or third-party options

Foreword

“

For years, data teams have been forced into an unnecessary tradeoff: build complex real-time pipelines or settle for slow, batch-driven analytics. But modern businesses need something more flexible: data that arrives exactly when it creates value. That's the promise of right-time data.

Snowflake has enabled this shift with a wide range of ingestion methods, but choosing, combining, and managing them has become a challenge in its own right. At Estuary, we set out to remove that complexity and give organizations a unified way to deliver data at the cadence each workflow demands.

This guide distills what we've learned from helping companies adopt right-time architectures. I hope it gives you a practical framework for designing Snowflake pipelines that are not just fast but also clear, efficient, and adaptable over the long term.



Dave Yaffe

CEO, Estuary



Foreword

“

Across companies I work with, one issue consistently holds back data teams: delivering the right level of freshness for each workflow. Some processes truly need sub-second updates; many don't. Yet organizations often overengineer or overspend in pursuit of “real-time” everywhere, or rely on batch when it can't keep up.

The shift to right-time data brings balance. It recognizes that different use cases demand different latencies, and that Snowflake's ecosystem now enables these choices, if teams know how to evaluate them.

This whitepaper provides a clear, practical framework to help leaders make smarter ingestion decisions grounded in business value rather than hype. It's precisely the kind of guidance the industry has been missing.



Ben Rogojan

Seattle Data Guy



01	INTRODUCTION	2
02	FOREWORD	4
03	TABLE OF CONTENTS	6
04	THE RIGHT-TIME DATA MANIFESTO	8
	Why Timing Is Now a Strategic Data Asset	10
	Right-Time Maturity Model	12
05	SNOWFLAKE	14
	Snowflake Ingestion Methods	15
06	THE POWER OF THE RIGHT SNOWFLAKE INTEGRATION AT THE RIGHT TIME	19
	A Framework for Choosing the Right Snowflake Ingestion Strategy	21
	The Hidden Costs of Choosing the Wrong Ingestion Method	23
	Total Cost of Ownership: Rethinking Data Integration Economics	24
07	DATA INTEGRATION GUIDES: NATIVE SNOWFLAKE	30
	Batch Workflows with COPY INTO	31
	Continuous Micro-Batches with Snowpipe	34
	Real-Time Workflows with Snowpipe Streaming	39
	Managed Ingestion with Openflow	42
	Quick Comparison Chart: Native Ingestion	48

08	DATA INTEGRATION GUIDES: THIRD-PARTY INTEGRATIONS	52
	Common Setup Steps	54
	Create Snowflake Resources	55
	Snowflake User Authentication	57
	Snowflake Setup with Airbyte	58
	Snowflake Setup with Estuary	59
	Connection Security with PrivateLink	61
	Connector Management with Infrastructure-as-Code	62
	Snowflake Setup with Fivetran	63
	Comparing Snowflake Integration Solutions	64
09	DATA INTEGRATION GUIDES: WHERE DOES YOUR DATA GO NEXT?	67
	Adding dbt to the Mix	68
	Using Data in Cortex or Other AI Use Cases	69
	Reverse ETL or Snowflake as a Source	70
10	ESTUARY	72
	Who We Are	73
	Estuary in Your Snowflake Stack	73
	Right-Time Integration with Estuary: A Strategic Advantage for Snowflake Customers	75
10	IN THEIR OWN WORDS: WHAT OUR CUSTOMERS SAY	77
11	APPENDIX: HELPFUL RESOURCES	80

The Right-Time Data Manifesto



The Right-Time Data Manifesto

After achieving effectively real-time data, with streaming solutions reaching millisecond-level latency, what's next for the data world?

We believe the answer is focusing on *right*-time data solutions. Real-time is an astounding feat, and necessary for responsive, time-sensitive workflows. But not every problem is a nail just because we've made a really powerful hammer.

The fact is, while real-time is indispensable for certain use cases, it's not how most of the business world experiences data. Weekly reports, quarterly reviews with daily granularity, and dashboards with hourly refreshes are all still relevant and widely used. In these cases, real-time data would in fact be inefficient, burning resources for no additional advantage or gain.

These use cases aren't wrong. That weekly progress report doesn't need to turn into a second-by-second review. Analysts don't need to be glued to a dashboard to make use of it.

The problem is, a lot of data rhetoric runs on real-time *versus* batch. One or the other. And often holds up real-time streaming as the ultimate goal.

Right-time, therefore, is about options. Real-time and batch don't need to be pitted against each other.

Both can work in harmony, and you can choose the right latency for each use case.

Snowflake is a clear example of the right-time data promise.

With numerous ways to ingest data into a central location for analysis, Snowflake can handle data at a range of latencies. The addition of super-fast Snowpipe Streaming doesn't mean that bulk loading has gone by the wayside.

All you need to do is know which data ingestion method matches your use case.

This guide will explore the various methods to get data into Snowflake, with tips on latency and use cases along the way, so you can choose the right method for your right-time solution.

[Learn More About The Right-Time Data Approach](#) →

Why Timing Is Now a Strategic Data Asset

Time has a longstanding tradition of being an important business asset, shown in phrases like “time is money.” First to market with a new product, business models built on saving consumers' time, and managing operations to save employees' time are all common pursuits.





Data is no different.

Timing for your data can determine speed-to-insight, risk exposure, and, of course, cost. And in the data world, less time often equals more money.

Increasingly, latency is no longer a question of whether engineering teams *can* achieve super-low latency, but whether they *should*. Essentially, latency should be treated as a business lever, with the same cost-benefit analysis that other business decisions undergo.

Going all-in on real-time can lead to overspend, while relying solely on batch can underserve. The wrong latency compounds into poor decisions based on outdated information or wasted compute. Rigid architecture and accumulating tech debt tied to specific solutions make it difficult to change latency decisions later.

In contrast, right-time data means ensuring that each department receives data at *exactly* the cadence that maximizes its business outcome. For example:

-  **Product:** Sub-second for personalization
-  **Finance:** Scheduled batch for predictable reconciliation
-  **RevOps:** Intra-day for forecasting
-  **Supply chain:** Seconds-to-minutes to mitigate disruptions

To embrace the right-time mindset, leaders should think of latency as a portfolio, where each stream deserves individual attention, ensuring it's right-sized and optimized. Taken together, this portfolio will likely be a diversified mix of latencies.

Right-Time Maturity Model

The general trajectory for most organizations will be from a batch-focused data model, gradually adding in lower latency capabilities, until reaching a point where the organization is comfortable with either and can focus on optimizing existing and new data streams for right-time architecture.



Consider these stages on the path to right-time data maturity:

1. Reactive (Batch-First)

- Weekly or daily updates
- Siloed ingestion

2. Operational (Micro-Batch)

- More frequent updates
- Faster dashboards

3. Real-Time Capable (Event-Driven)

- Ability to stream data for certain use cases
- Some latency mismatch and complexity

4. Right-Time Intelligent Architecture

- Unified platform like Estuary
- Latency chosen per use case
- Lowest-cost, highest-flexibility model
- Fully AI-ready*

Organizations can streamline their transition through the stages, reducing technological friction as they grow, by choosing a flexible platform upfront that can grow and unlock new capabilities with them.

Snowflake



Snowflake

Overview

Snowflake is one of the premier data warehouses in use today. Cloud-agnostic and able to work with an array of structured and unstructured data, Snowflake provides flexible options to aggregate and analyze your data.

Snowflake gracefully handles high volume, complex queries, and provides flexible options to handle data exactly how you need.

Snowflake Ingestion Methods

Snowflake offers several native ways to ingest data into your warehouse. Each method has different latency and other considerations. We'll explore these methods in further depth, including how to ingest data using each of them, in the sections on data integration.

In the meantime, it's useful to understand the options at a high level as we continue discussion of right-time data in the context of Snowflake integration.

01. Batch/COPY INTO Ingestion

Snowflake's batch COPY INTO command merges data from a staging source into your Snowflake table.

The simplest ingestion option to load data into Snowflake, COPY INTO is often useful for ad hoc data loads. Automating it into a robust pipeline, however, can be more effort.

02. Snowpipe

Snowpipe is a micro-batch method of ingesting data. Pipes can be set up with COPY INTO commands that run continuously.

Snowpipe offers lower latency than standard COPY INTO ingestion, but its minutes-long delay still may not be optimal for truly real-time use cases.

03. Snowpipe Streaming

The lowest-latency way to load data into Snowflake, Snowpipe Streaming uses a row-based ingestion method rather than copying files into your tables. It can also require the most complex integration unless using pre-built solutions like Estuary.

Snowpipe Streaming can also be broken down into Classic and High Performance variations. These use different SDKs, different pricing models, and include some other variations, so it can be important to know which version of Snowpipe Streaming you're using.

04. Openflow

Snowflake Openflow is Snowflake's built-in, NiFi-powered data integration service that lets users visually build pipelines to ingest, move, and lightly transform data directly within the Snowflake platform.

Announced after Snowflake's acquisition of Datavolo, it supports a wide range of sources, real-time and batch workloads, and flexible deployment options (Snowflake-hosted or BYOC).

While these are the main Snowflake-native methods to load data into your warehouse, there are also other connectors and integrations you can use for interfacing with third-party data and systems.

While a single guide would be hard-pressed to cover every option in-depth, here are a few more ingestion options of note:

01. Snowflake Kafka Connector

Users who already support an Apache Kafka ecosystem can use [Snowflake's Kafka connector](#) to stream Kafka messages into Snowflake tables.

The Kafka connector uses Snowpipe or Snowpipe Streaming as the underlying data ingestion methods.

02. Other Native Connectors

Snowflake native SaaS and database connectors offer turnkey integrations that automatically ingest and continuously update data from popular SaaS apps and databases directly into Snowflake.

With built-in support for historical loads and incremental syncs, these connectors simplify data integration for sources like Google Analytics, Looker Studio, ServiceNow, SharePoint, MySQL, and PostgreSQL, eliminating manual API work and accelerating analytics.

03. Snowflake's Data Integration Ecosystem

Snowflake has a whole ecosystem of data integration partners, covering a range of ingestion options.





We'll discuss a few notable integrations later on. You can also see [Snowflake's full list of third-party data integrations](#).

The Power of the Right Snowflake Integration at the Right Time



The Power Of The Right Snowflake Integration At The Right Time

When deciding between integration options, whether native or third-party, it is helpful to have an idea of the most important criteria for your use case. This can include, for example:

-  **Functionality:** How will the integration work? Besides setup time, will it require maintenance? Does your use case require top-of-the-line reliability?
-  **Latency:** Going back to the right-time concept, when do you need your data? As soon as technologically possible? In recurring batches? Might these requirements change in the near future?
-  **Cost:** How much can you afford to spend on your data integration? Besides any third-party integration costs, will you need to spend engineering hours on setup and maintenance? Does a third-party integration use Snowflake resources efficiently?
-  **Support:** If something breaks, how swift are support turnaround times? Are support personnel well-informed? Does your actual experience match up with any stated SLAs?

- ☰ **Specific features:** Do you have certain requirements, such as enterprise-grade security, or compliance requirements to review configuration changes before deploying?

It can be difficult to find a solution that perfectly matches every criterion, so you may want to rank them or sort them into buckets, such as “must haves” and “nice to haves.”

Some criteria are also difficult to determine upfront, such as how reliable the product is or how knowledgeable the support team is. It’s therefore often a good idea to perform a PoC, or proof of concept, to test the fit of the solution.

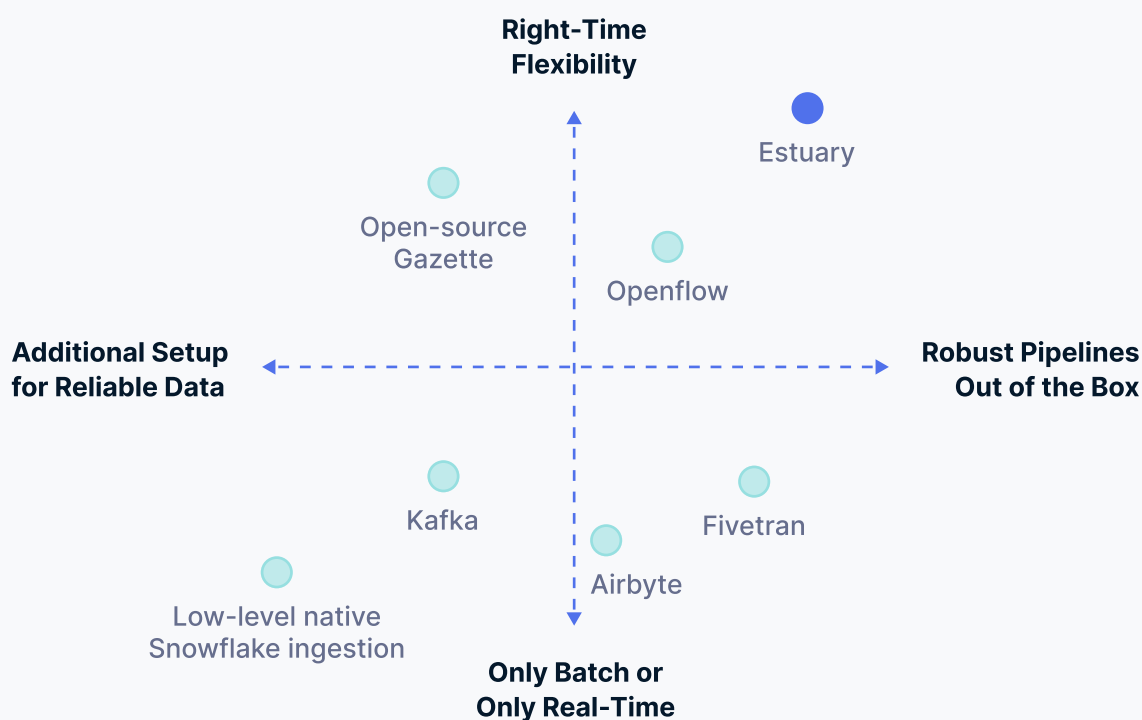
Most data movement platforms provide free trials. If you’re interested in a higher customer tier than is generally associated with the free trial, such as one that offers special deployment options, you can check about your options with the sales team.

This may seem like extra effort for a solution you’re not sure about yet, but a little time at this stage can save a lot of resources and headaches later down the line. The wrong solution can quickly rack up costs and engineering time or, in the worst cases, result in lost data.

A Framework for Choosing the Right Snowflake Ingestion Strategy

One way to condense all the questions surrounding the right fit for an integration is with a quadrant graph.

Take two of the most important aspects for your solution and dig into how the various options rank on each axis.



Here, we plot out options based on built-in right-time capabilities, as well as how much effort is required to achieve enterprise-ready robust and stable pipelines.

Options that can only handle batch or streaming data rather than flexibly choosing between the two fall on the lower half of the y-axis. This includes Snowflake's native ingestion options, such as bulk COPY INTO (which is strictly batch) and Snowpipe Streaming (only real-time).

The relative work required for an enterprise-grade solution is displayed along the x-axis, with open-source solutions generally

requiring more setup and tweaking to fit specific use cases than robust platforms that provide durable pipelines out of the box.

While these are common requirement criteria, similar analysis can be performed for other important factors, such as whether integration options support specific features, offer flexible deployments, or how they rank on security measures.



The Hidden Costs of Choosing the Wrong Ingestion Method

One Snowflake integration method is about as good as any other, right?

Wrong.

Choosing the wrong type of ingestion for your use case, hacking together a faulty solution, or deciding on an inefficient third-party option comes with numerous costs, not all of them obvious.

As you review the ingestion options presented in this guide, keep your use case in mind to avoid unnecessary costs. Any misalignment in your ingestion strategy can create:

-  **Snowflake credit waste:** Running your warehouse too often or too long, or over-provisioning your warehouse size
-  **Engineering cost bloat:** Custom integrations for real-time data or modifying third-party solutions to work for your use case rack up engineering costs quickly

- ☰ **Opportunity cost:** Slow time-to-insight can lead to missed opportunities in fast-moving industries
- ☰ **Inconsistency and unreliability:** Setting up multiple integrations to cover multiple use cases can result in data mismatches, or downtime might result in missing data, causing stakeholders to lose trust in the organization's data
- ☰ **Architectural rigidity:** Tightly coupling systems with a certain ingestion type can make it difficult to pivot when requirements change, causing friction when trying to implement AI, streaming, or other use cases later down the line

The part that makes this so tricky is that there isn't a one-size-fits-all answer. Batch overuse can cause inefficiencies and stale data, while real-time overuse can result in runaway compute or heavy operational burdens, such as managing Kafka.

Overall, organizations can inadvertently **overspend by 2-5x** when solving for the wrong timing. As this may not be apparent upfront, these sorts of mistakes can take years to correct.

That's why it's so important to solve for *right-time* data from the start. By focusing on right-time usage, you can simplify pipelines, decrease Snowflake credit spend, and scale more predictably.

Total Cost of Ownership: Rethinking Data Integration Economics

When choosing your data ingestion solution for Snowflake, the solution's cost is more complex than just the platform's vendor cost.

You can actually break “cost” down into several rough categories. All of them combined are the solution’s total cost of ownership.

As mentioned, there is the obvious **vendor cost**. This is the platform-specific pricing you’ll see in your monthly or annual bill. Even within this one category, it can be difficult to compare solutions: pricing strategies range from straightforward per-GB throughput costs to tricky-to-calculate monthly active rows (MAR) to credit-based pricing like Snowflake itself that can depend on the specific service or feature you’re using.

There are also your in-house **engineering or administrative costs**. Free, open-source, or other DIY integrations may sound appealing from a vendor cost perspective, but costs quickly add up when you figure in engineering setup and maintenance. Is a custom solution worth the time and energy of your in-house engineers?

These engineering and administrative costs may seem like more of a problem for these free or DIY integrations, but you should also watch out for it with paid data integrations. Is a third-party platform actually easy to use? Does it do what you need it to out of the box, or will you end up cobbling together your own custom hack to get it to work for you?

Some costs may even be a complete unknown upfront, such as **connected services costs**. Since Snowflake provides so many offerings, Snowflake has a reasonably complex cost structure. You may be able to predict your monthly cost with them based on the services you use, but when you integrate using a third party, it can be hard to tell how efficiently that party will use your Snowflake resources.

For an example, consider using an open-source connector that's freely available on GitHub. Besides the engineering and maintenance costs, there's also the connector's performance to consider. It may get the job done, shuttling data into Snowflake, but without efficient usage refinements. Each sync may end up reprocessing data or using an inefficient query that takes longer than it really needs to, racking up Snowflake compute costs.

Compare that with a dedicated high-quality connector that a team of experts meticulously improves. Seconds of compute get shaved off where possible. The latest Snowflake offerings get implemented to improve cost and latency. This integration eventually significantly outperforms the free one, resulting in equally significant Snowflake savings.

To help illustrate this *entirely hypothetical* use case, consider this side-by-side comparison. [Headset](#) ran two Snowflake integrations and their respective warehouses simultaneously while migrating their data pipelines. Transferring the same data over the same timeframe, Headset found that Estuary used **75% fewer** Snowflake credits compared to Airbyte.

It's not enough for a data integration to get data where it's supposed to be going. It needs to do so at the right time, for the right cost, with efficient usage of the right resources in connected services.







All of these costs, especially the ones that aren't neatly laid out in a pricing table upfront, add up to another strong reason to start with a proof of concept. Trialing the product is the only real way to uncover some of these underlying costs.

Finally, there is the even more nebulous **opportunity cost** of choosing the wrong solution or building on the wrong architecture. This can be particularly acute when compounded by *vendor lock-in*, but can occur with any tightly coupled architecture.

Consider a system built on the assumption that all data is batch, with data analysts as expected users. The market and requirements change, and the organization wants to add a new feature to the product. The new feature would require the same data to be rerouted in a clean, succinct form back to the end consumer, in real-time. But since the system was built fully around batch, reworking it to handle real-time data would require a lot of dedicated engineering time.





If the new feature is scrapped as out of scope, that's a lost opportunity. If engineers are taken away from other tasks to work on it, that's lost time on those other features and improvements. Either way, the rigidity of the initial decision has incurred additional costs down the line.

So, you can ultimately break TCO down into these categories:

-  Data integration platform or vendor costs
-  Engineering and maintenance required for the solution
-  Connected service costs, such as Snowflake compute and storage
-  Opportunity cost




While it can be difficult to estimate some of these costs, it's important to consider what they might include, and even formulate some worst-case scenarios, to avoid costly surprises.

A fully right-time service or platform can help you manage these costs by:

-  Managing multi-latency pipelines in one place, without needing to track and maintain multiple integrations
-  Lowering credit usage by only using the credits you need for each use case
-  Eliminating redundant tooling and other tech debt
-  Allowing you to update latencies at a moment's notice when requirements change

Altogether, moving to a right-time model can result in 60-80% reduction in ingestion costs, improved SLAs, and confidence in your data structure.

How can we quantify this cost reduction when TCO includes so many uncertainties? By seeing it in action. Besides the side-by-side Headset example, we routinely talk with customers who have seen significant differences in their spend when they start transitioning to right-time data:

-  [Connect&GO](#) saw 5x lower vendor costs while lowering latency by 180x
-  [Livble](#) reduced both their platform costs and Snowflake spend
-  [Xometry](#) lowered total data integration costs by 60%

Managing spend isn't just a pipe(line) dream: we've found it to be eminently achievable by focusing on efficiency in data movement.

Now that we've discussed the theory, in the following sections we'll take a closer look at how to actually implement Snowflake data ingestion and explore trade-offs between specific solutions.

Data Integration Guides: Native Snowflake



Data Integration Guides:

Native Snowflake


It's no wonder that as a top cloud warehouse, Snowflake provides a range of options for getting data into your Snowflake instance.

While Snowflake also offers [connectors](#) to integrate with specific systems, this section of the guide will focus on Snowflake's main [data loading options](#): bulk loading, Snowpipe, Snowpipe Streaming, and the new Openflow.

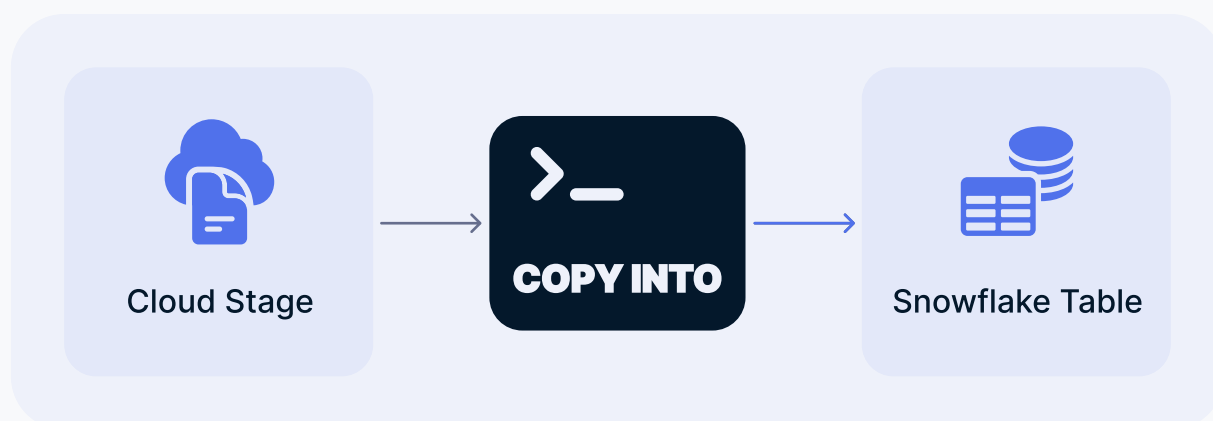
Batch Workflows with COPY INTO

Bulk import, bulk loading, or COPY INTO: whatever you call it, this method of data ingestion falls firmly into the batch processing category. This can be a simple data integration to start with, and can be useful for ad hoc or infrequent additions, but it still carries some caveats and requirements to be aware of.

Essentially, the COPY command copies information from one location to another. Data must therefore already be in a table or other staging location before you can copy it to Snowflake. This can look like:

-  Uploading local files to Snowflake by first PUTting the data into a Snowflake stage.

- ☰ Referencing an existing external stage in cloud provider storage, such as AWS S3 or Google Cloud Storage.



Besides the staging location, you will also need to set up and manage Snowflake virtual warehouse resources. This virtual warehouse should be sized appropriately and suspended when not needed for efficient usage.

You can then use the COPY INTO command like any SQL query, running it in Snowflake's SQL editor.

For example, this simple COPY INTO query copies data from CSV files in an AWS S3 bucket, providing credentials to access the S3 bucket:

```
COPY INTO snowflake_table
  FROM s3://aws-bucket/path'
  CREDENTIALS = (AWS_KEY_ID='your-aws-key-id'
  AWS_SECRET_KEY = 'your-aws-secret-key')
  FILE_FORMAT = (TYPE = 'CSV');
```

Attaching credentials to each query like this isn't ideal, so you can instead create and use a [storage integration](#). This lets you store a cloud provider IAM entity that you can use for your external cloud stage.

The storage integration can also specify multiple locations to support multiple external stages.

Using a storage integration with the previous COPY command would look like:

```
COPY INTO snowflake_table
  FROM 's3://aws-bucket/path'
  STORAGE_INTEGRATION = your_integration_name
  FILE_FORMAT = (TYPE = 'CSV');
```

Find a full list of COPY INTO options in [Snowflake's reference](#).

There is, of course, a difference between batch loading and manual loading. While these examples showcase manual SQL queries, you can set up your own integration or choose from a number of third-party integrations to set a scheduled cadence for this batch option. These integrations make Snowflake's COPY INTO command a convenient batch solution.

Continuous Micro-Batches with Snowpipe

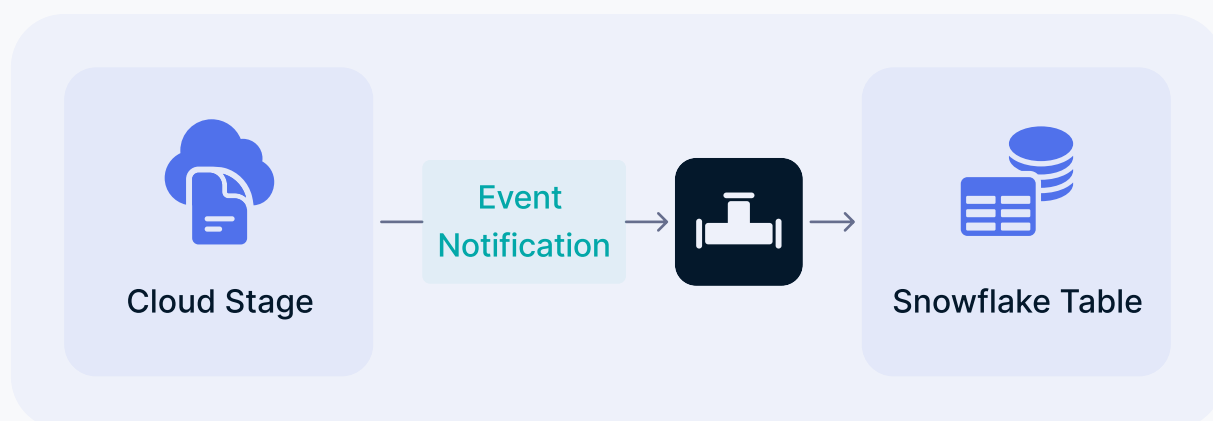
One option for automating COPY INTO commands is to use **Snowpipe**. Snowpipe's standard mode of operation is to automatically ingest new data as it lands in a stage in micro-batches. This takes us out of the realm of scheduled batches while not quite being real-time: Snowpipe latency tends to be in the several minute range.

Besides latency, Snowpipe has some other differences from bulk loading:

- ☰ Snowpipe uses serverless resources rather than a user-defined warehouse
- ☰ Because of this, Snowpipe uses a different cost structure, based on serverless compute or data volume depending on Snowflake edition
- ☰ Snowpipe loads may be split into multiple transactions, unlike bulk loading which always performs loads as a single transaction

In other ways, Snowpipe remains similar to bulk loading.

Data can be loaded in files from an intermediate stage and you must define a COPY statement for your pipe.



To set up a Snowpipe integration, you will need to:

- 01. Create a pipe object**
- 02. Specify the desired COPY statement for the pipe**
- 03. Define a mechanism to detect when new files are available**

New files can be detected either via event notifications from the cloud staging location or by setting up a REST integration.

For example, you can create a simple pipe using the following SQL syntax:

```
CREATE PIPE pipe_name
AS
COPY INTO snowflake_table
FILE_FORMAT = (TYPE = 'CSV');
FROM @stage
```

Each cloud platform will have its own requirements or options to set up cloud messaging. For AWS S3, one option is to use SQS or Simple Queue Service. You can set up these notifications from your staging bucket:

01. In your AWS dashboard, select the storage bucket you want to use for staging.
02. Under **Properties**, find the **Event Notifications** section.
03. Select **Create event notification**.
04. Fill out event notification details, such as:
 - **Name:** Provide a descriptive name for the event notification
 - **Events:** For Snowpipe, you want to receive notifications when new files are available, so choose the **ObjectCreate (All)** option
 - **Send to:** Choose **SQS Queue** as your destination
05. Save changes.

You can indicate that a pipe should use automatic ingestion options by adding the `AUTO_INGEST` option to the pipe definition. For example:

```
CREATE PIPE pipe_name
  AUTO_INGEST = TRUE
  AS
  COPY INTO snowflake_table
  FROM @stage
  FILE_FORMAT = (TYPE = 'CSV');
```

Find instructions for working with other Amazon options like SNS (Simple Notification Service) or other cloud platforms in [Snowflake's docs](#).

You may notice that this type of event-based auto-ingestion is not required. If you want to use Snowpipe for scheduled batches, you can set up an integration to use the Snowpipe REST API. This allows you to notify Snowflake of new files for ingestion at a cadence of your choosing.

Essentially, a REST API integration boils down to sending a POST request to Snowflake's `insertFiles` endpoint at:

```
https://{account}.snowflakecomputing.com/v1/data/pipes/{pipeName}/  
insertFiles
```

The request body will need to contain the file paths for newly available files:

```
{  
  "files": [  
    {  
      "path": "storagePath/file1.csv"  
    },  
    {  
      "path": "storagePath/file2.csv"  
    },  
  ]  
}
```

Each file object can also include a size field to help improve performance.

This DIY Snowpipe integration isn't as straightforward as it may seem: to create a robust connection, you would need to manage new file detection, implement the API integration including how to handle non-200 responses, and determine what should happen if your pipeline breaks or parts of it experience downtime.

Real-Time Workflows with Snowpipe Streaming


On the other end of the spectrum from scheduled batches, we have real-time streaming. Real-time can be notorious for being complex and difficult to manage. Snowflake's Snowpipe Streaming doesn't completely disabuse that notion.

Snowpipe Streaming requires implementing an SDK or API. There are also two different Snowpipe Streaming options to choose from: an older, but potentially more established, Classic version, as well as a newer High Performance version. Each has distinct requirements and architectural considerations.

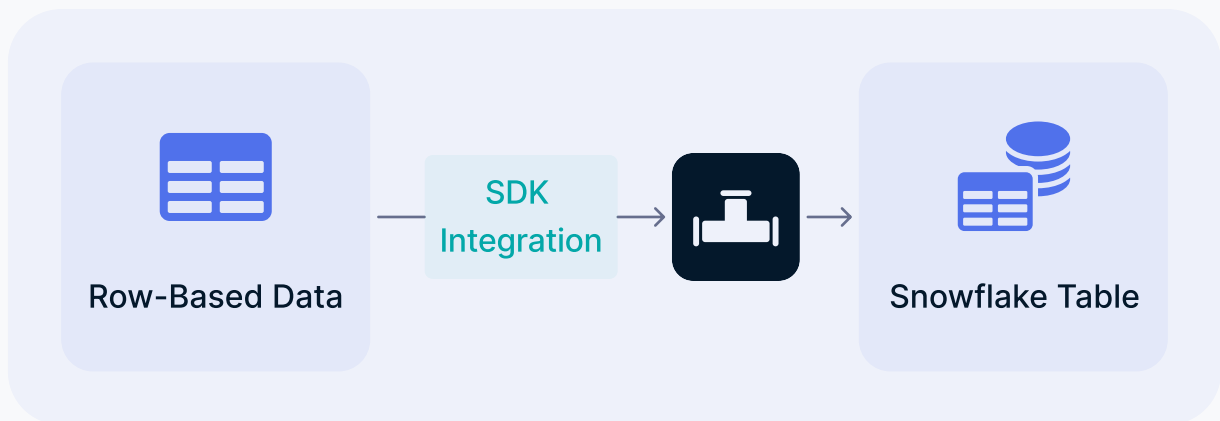
For example, Classic can support earlier Java versions than High Performance, but only High Performance offers a Python SDK. High Performance utilizes a Snowflake pipe object while Classic bypasses that requirement. Each has their own unique [limitations](#) and [considerations](#).

So, while Snowpipe Streaming is often lumped into its own category to differentiate it from bulk loading or regular Snowpipe, it can also be seen as two separate data ingestion options.

Despite this, there are some aspects that the two versions of Snowpipe Streaming share:

-  **Low latency:** Whichever Snowpipe Streaming version you use, latency is down to the second range, allowing for real-time use cases. High Performance, as the name suggests, may be better able to handle higher throughput use cases, up to 10 GB/s.

- ☰ **Row-based data:** Snowpipe Streaming streams at the row or message level rather than loading data in files.
- ☰ **No intermediate staging step:** Data doesn't need to land in a staging location before being ingested.
- ☰ **Serverless resources:** Both versions of Snowpipe Streaming are serverless options, so you don't need to keep a virtual warehouse running for compute resources.



To simplify matters somewhat, Snowflake recommends going with High Performance for new streaming use cases, so we'll look at that version for our example.

As with Snowpipe, High Performance Snowpipe Streaming will require a pipe object in Snowflake, which in turn requires a COPY statement. In this case, though, we're copying data from a streaming source rather than a specific staging location.

The pipe declaration will therefore look something like:

```
CREATE OR REPLACE PIPE pipe_name
  AS COPY INTO snowflake_table
  FROM (SELECT $1, $1:c1, $1:ts FROM TABLE(DATA_SOURCE(TYPE
    ⇒ 'STREAMING')))
```

Note that this declaration can define specific columns to select from your streaming source. You should be familiar with your data structure and Snowflake advises implementing schema validation and error recovery as [best practices](#).

To use the Snowflake SDK, install the snowpipe-streaming dependency. If using Python, for example, you can use pip:

```
pip install snowpipe-streaming
```

Your integration will need to be able to authenticate with your Snowflake account. You can create a key pair for secure JWT authentication. See the section below on third-party integrations for instructions on setting up a key pair.

Provide this information in a profile.json file or directly in your implementation code.

```
{  
  "account": "your_account_identifier",  
  "user": "your_username",  
  "url": "https://your_account_identifier.snowflakecomputing.com:443",  
  "private_key_file": "rsa_key.p8"  
}
```

The main implementation code will then need to:

- ☰ Create a `StreamingIngestClient` with your profile, pipe, and database information
- ☰ Open a *channel* for data ingestion
- ☰ Use the channel to append the correct data from incoming rows to your table
- ☰ Handle any [errors](#) that occur

Snowflake [provides simple example implementations](#) for Java and Python to get you started. Schema information, validation, and how detailed you want your error handling to be will be down to your unique use case.

Managed Ingestion with Openflow

So far, all of these native Snowflake options feel fairly low-level: good building blocks, but requiring some extra work to get a robust data ingestion pipeline up and automated.

To address the need for simpler, streamlined, more managed data ingestion, Snowflake has introduced Openflow.

Openflow is Snowflake's integration service, allowing you to visually create pipelines directly in your Snowflake dashboard. Built on Apache NiFi and accessed through Snowsight, Openflow allows you to manage deployments and runtimes for your pipelines.

Essentially, this provides a native way to automate bulk load, Snowpipe, or Snowpipe Streaming data ingestion into Snowflake.

As a fairly new offering, Openflow currently includes around twenty connectors across SaaS, database, and streaming systems

One caveat to keep in mind: Openflow deployments are currently limited to AWS and Azure regions, or just AWS regions, depending on deployment type.

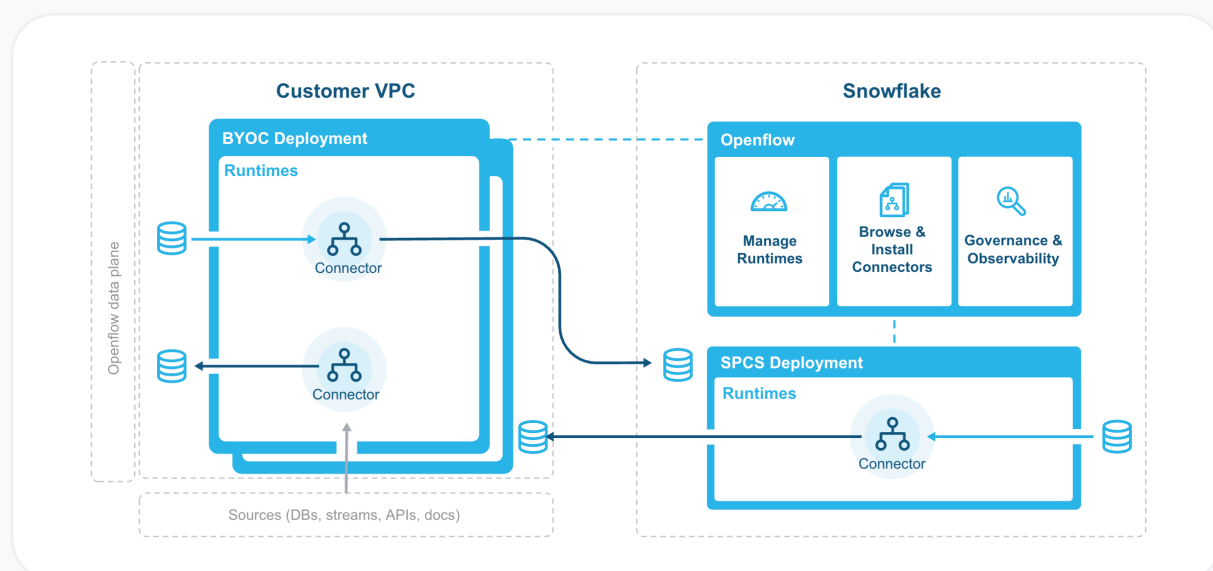


Image source: snowflake.com

To start setting up an Openflow project, your Snowflake account will need to have Openflow enabled with a role that has privileges to create Openflow projects and connections. You may also need a Snowpark Container Services runtime or BYOC data plane if you wish to use these options with your integration.

STEP 1 Create an Openflow Project

01. In Snowsight, go to Data » Openflow.
02. Click **Create Project**.
03. Give your project a name and choose the Control Plane.
04. Select your Data Plane (Snowflake-managed or BYOC cluster).
05. Click Create: this sets up the isolated workspace for your ingestion flows.

STEP 2 Configure a Source Connection

01. Inside your project, open the Connections tab.
02. Click **New Connection** and choose a connector (e.g., PostgreSQL, Kafka, Salesforce, S3).
03. Provide required connection details:
 - a. Host, port, database, user/password (for databases)
 - b. API tokens (for SaaS sources)
 - c. Bucket/container paths + credentials (for storage)

04. Test the connection.
05. Save it: Openflow exposes it as a reusable endpoint in your flow canvas.

STEP 3 Build a Data Flow in the Canvas

01. Navigate to the Flows tab and create a new flow.
02. On the NiFi-style canvas:
 - a. Drag the Source Processor that corresponds to your connector (e.g., QueryDatabaseTable, ConsumeKafkaRecord, FetchFile).
 - b. Add intermediate processors if needed:
 - i. Record transformation (schema evolution, filtering, routing)
 - ii. Format conversion (CSV → JSON, Avro → Snowflake variant)
 - iii. Error handling (RouteOnError → DLQ)
03. Add a **Snowflake Write processor**, such as:
 - a. PutSnowflake (bulk COPY)
 - b. PutSnowflakeStreaming (Snowpipe Streaming)
04. Connect the processors with flow relationships.

STEP 4 Configure Snowflake Target Settings

For the target processor, specify:

- 01.** Snowflake connection
- 02.** Target database, schema, and table
- 03.** Load mode:
 - a.** Bulk Load (COPY) for large batches
 - b.** Snowpipe for micro-batch
 - c.** Snowpipe Streaming for real-time ingestion
- 04.** Optional:
 - a.** Merge keys
 - b.** Error handling behavior
 - c.** Auto-create table (if supported by the processor)

STEP 5 **Validate and Start the Flow**

- 01.** Click **Verify Flow** to check for missing or incompatible processor configs.
- 02.** Start the flow by toggling Run on the root process group.
- 03.** Watch the initial run in the Monitoring panel.

STEP 6 **Monitor, Scale & Troubleshoot**

Monitoring

- 01.** Use Openflow Monitoring to view throughput, scheduling, errors, and flowfile counts.
- 02.** Inspect processor queues for backpressure or bottlenecks.

Scaling

- 01.** Increase parallelism by adjusting processor Concurrent Tasks.
- 02.** Scale out data plane nodes (Snowpark containers or Kubernetes replicas).

Troubleshooting Tips

- 01.** For schema mismatches: add a ConvertRecord / UpdateRecord processor.
- 02.** For slow loads: switch to Snowpipe Streaming or add threading.
- 03.** For detailed logs: access the NiFi logs from the data plane (K8s or Snowpark container logs).

Limitations

While Openflow helps automate the basic building blocks of Snowflake's low-level data ingestion options like bulk load through Snowpipe Streaming, the new offering still has a number of limitations that in turn can limit the usefulness of this solution. In general, you will not be able to solve all your data movement needs solely with Openflow.

- ☰ **Few source connector options:** There are currently only about 20 native connector options, compared against more established pipeline platforms' hundreds. You may not be able to transfer data from all your data sources.
- ☰ **Limited cloud provider support:** Openflow Snowflake deployments are currently limited to AWS and Azure regions, while BYOC deployments are only available for AWS.
- ☰ **No reverse ETL:** Openflow currently only supports Snowflake as a destination. You can't move data to other platforms or back out of Snowflake once it's cleaned and analyzed.
- ☰ **Limited transformation options:** The drag-and-drop UI is useful for defining general data flows, but doesn't handle mappings between fields.
- ☰ **Limitations in individual connectors:** Make sure to read connector docs carefully. Some connectors do not support key functionality, such as schema evolution.

With that, we've covered the main Snowflake data ingestion types that others are built on top of. But the question remains: which one is right for you? Or does it seem like none of them fit perfectly? Third-party integration options can help even out the lumps, so read on before deciding for certain.

Quick Comparison Chart: Native Ingestion

Before setting up an integration with Snowflake, you should know which ingestion method works best for your use case.

Each ingestion method has its own associated uses, latency, and costs. While some have greater complexity, like Snowpipe Streaming requiring SDK implementation, pre-built integrations from third parties can level the playing field. We'll explore pre-built integrations in the next section.

Use the following table for a quick comparison:

	Bulk COPY INTO	Snowpipe	Snowpipe Streaming	Openflow
Use Case	Ad-hoc data ingestion for one-time or infrequent transfers	Continuous, but not instantaneous, data replication	Real-time data replication	Unified ingestion for batch, micro-batch, and streaming from ~20 sources
Setup	Manual run of COPY INTO command	Create a new Pipe with a defined COPY INTO statement; automate with REST APIs or cloud messaging	Custom application that implements one of two Java SDKs or REST APIs	Create Openflow project and drag-and-drop pipeline using connectors

	Bulk COPY INTO	Snowpipe	Snowpipe Streaming	Openflow
Complexity	Theoretically as simple as COPY INTO <table> FROM <source>; with additional complexity from options, compute setup, and staging setup	You don't need to worry about provisioning compute resources, but you do need to define a way for the Pipe to receive notifications of new data	More complex; requires robust Java, Python, or REST development and additional configuration	Moderate; low-code but requires basic NiFi concepts
Latency	Deterministic	Half-minute to minutes	Sub-second/ millisecond latency	Configurable: batch to real-time depending on processor
Granularity	Batch files	Micro-batches	Row-level streaming	Files, micro-batches, or event streams
Compute	Requires a user-provided and sized virtual warehouse	Utilizes serverless Snowflake compute resources	Utilizes serverless Snowflake compute resources	Snowpark Containers (serverless) or BYOC Kubernetes

	Bulk COPY INTO	Snowpipe	Snowpipe Streaming	Openflow
Staging Storage	Requires extra data storage for staging area	Requires extra data storage for staging area	Does not require staging, streamlining architecture	Optional; depends on pipeline design
Costs	Compute costs plus warehouse management costs	Compute costs plus per-file charge (Snowflake announced this would change to a per-GB charge)	Depends on SDK used; throughput/per-GB cost or compute costs plus client connection costs	Snowpark container credits or BYOC infra + Snowflake ingestion costs

The right option for your use case may actually be multiple data ingestion methods.

For example, you may want to implement real-time analytics for a specific subset of data and route the results back to your main application for an enhanced user-facing experience, while the bulk of your data syncs on set schedules for your data team to work with.

If Openflow doesn't fit your use case, either not supporting your required sources or your cloud provider, this may sound like double the work, with multiple implementations to keep track of and double the things that can go wrong. It doesn't have to be that way. Here's where third-party integrations can really shine.

Data Integration Guides: Third-Party Integrations



Data Integration Guides: Third-Party Integrations




Snowflake has a thriving ecosystem of third-party integrations based off of Snowflake's native ingestion functionality.

Why might you choose to go with a third-party option?

Snowflake's native options of bulk loading, Snowpipe, and Snowpipe Streaming provide great building blocks, but those building blocks are very general, designed for as wide a range of use cases as possible. They can require fairly technical implementations, such as when setting up a Snowpipe Streaming SDK or even if you want to set up a durable, repeated version of bulk loading with COPY INTO.

Alternatively, you may appreciate Snowflake Openflow's ease of use, but find it too limited, needing more connector or deployment options.

When adding third-party integrations to the mix, you add the opportunity for:

-  Streamlined implementation
-  Opening up integration possibilities for less technical team members
-  Removing the integration maintenance burden from your team
-  Easy customization options

The array of integration options can make it easier than ever to get the timing on your data loads just right, whether needing real-time streaming or scheduled batches on predictable cadences. And the simplified setup of third-party integrations can make it more feasible to load some data as batches, some as streaming, without needing to maintain multiple implementations yourself.

On the other hand, the array of integration options means there are a lot of options to sort through.

We'll include guides on using a few of the top third-party integrations on the market, covered alphabetically, and then provide a succinct comparison of the platforms. You can also review [Snowflake's complete listing of third-party ETL integrations](#).

The guides and comparison in this section will focus on Snowflake as a destination. Each platform will have its own specifications and requirements for capturing data from source systems as well. Make sure that any integration you're considering supports the source systems you want to transfer data from.

Common Setup Steps




Before we dive into third-party integration guides, let's consider what we may need in Snowflake.

Most integrations require common Snowflake resources and use common authentication methods with Snowflake. Many solutions will require these resources, or a close variation.

Create Snowflake Resources

Because third-party integrations build off of Snowflake's native capabilities, these integrations will generally require you to create Snowflake resources in advance.

These resources generally include:

-  A user and role with required permissions
-  A database, with associated schema, for data load
-  A virtual warehouse to manage compute

The third-party integration will then take the user credentials to create and manage tables for you.

You can create these resources in Snowflake with CREATE statements in Snowflake's SQL editor. See the following examples for common usage.

Check any documentation for the integration: some platforms provide full example scripts for you to run or have different requirements.

Create a role

```
create role if not exists 'your-role';  
grant role 'your-role' to role SYSADMIN ;
```

Create a database and associated schema

```
create database if not exists 'your-db';  
use database 'your-db';  
create schema if not exists your-schema;
```

Create a warehouse with auto-suspend

```
create warehouse if not exists 'your-wh'  
warehouse_size = xsmall  
warehouse_type = standard  
auto_suspend = 60  
auto_resume = true  
initially_suspended = true;
```

Create a user with the defined role

```
create user if not exists 'your-user'  
default_role = 'your-role'  
default_warehouse = 'your-wh';  
grant role 'your-role' to user 'your-user';  
grant all on schema 'your-schema' to 'your-role';
```

Grant additional role permissions

```
grant USAGE
on warehouse 'your-wh'
to role 'your-role';
grant CREATE SCHEMA, MONITOR, USAGE on database 'your-db'
to role 'your-role';
```

Snowflake User Authentication

Once you create these resources, you may also need to perform some additional setup to allow your integration to work with your user. One common method is to set up a JWT key-pair.

You can generate a new key pair for JWT in a terminal session using openssl:

```
# generate a private key
openssl genrsa 2048 | openssl pkcs8 -topk8 -inform PEM -out
rsa_key.p8 -nocrypt
# generate a public key
openssl rsa -in rsa_key.p8 -pubout -out rsa_key.pub
# read the public key and copy it to clipboard
cat rsa_key.pub
```

You will attach your public key to your Snowflake user. Your integration will hold onto the private key for authentication.

Back in Snowflake, alter your integration user to use your newly-generated public key. Run the following in the Snowflake SQL editor, pasting in your own key:

```
alter user 'your-user' set RSA_PUBLIC_KEY= 'your-public-key';
```

Snowflake Setup with Airbyte




Airbyte is a popular open-source data pipeline platform offering a wide range of connector options.

Airbyte's Snowflake integration focuses on the bulk loading data ingestion method.

Airbyte Configuration

Airbyte uses the Snowflake resources mentioned in the “Common Setup Steps” section. Make sure these are created before you start configuring your Airbyte integration. You can also find a complete example setup script in [Airbyte's docs](#).

To start setting up in Airbyte, create a new Snowflake destination from the Airbyte dashboard. You will then need to provide these configuration details:

-  Snowflake **Host** domain
-  User details, such as **username**, **role**, and **password** or **private key**
-  **Warehouse**, **database**, and **schema**

You can choose whether your integration uses full refresh or incremental sync modes. Incremental sync modes require a specified *cursor* field.



Snowflake Setup with Estuary

Estuary is the right-time data platform, built for both sub-second speed and predictable schedules. You can create dependable ETL or ELT pipelines with automatic schema evolution, flexible deployment options, and more.




Estuary can support any of Snowflake's underlying data ingestion methods, from bulk loading to Snowpipe or Snowpipe Streaming. This makes it a flexible option when considering data latency and right-time requirements.

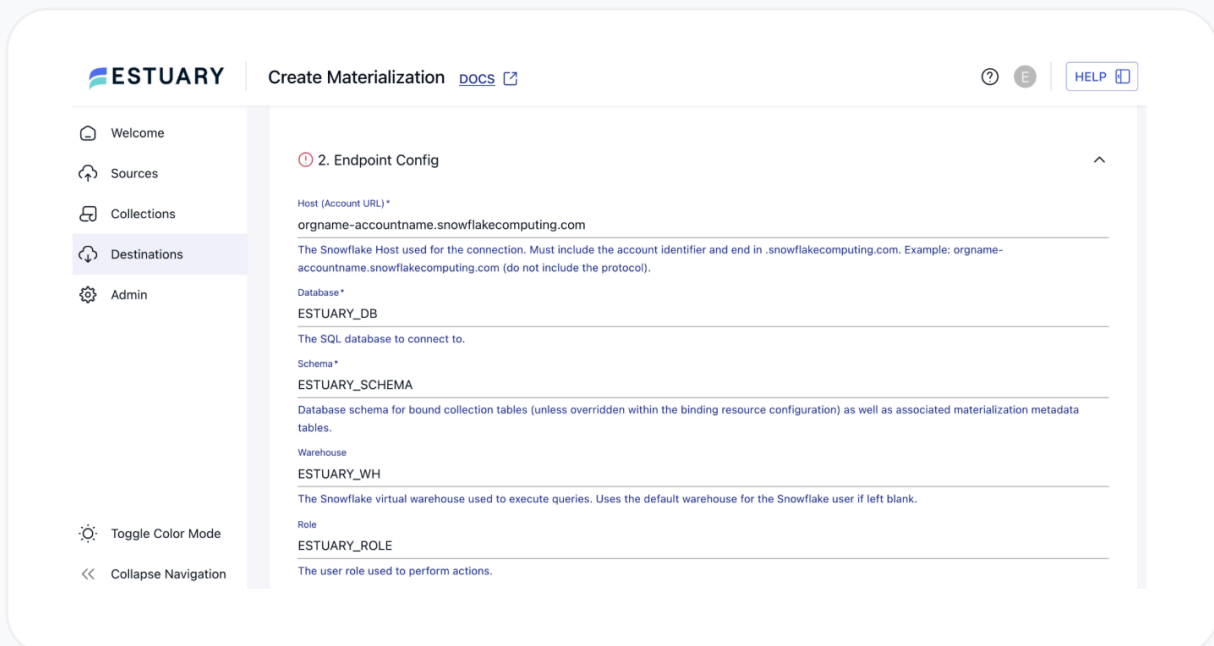
Estuary Configuration

Estuary uses the Snowflake resources mentioned in the “Common Setup Steps” section. Make sure these are created before you start configuring your Estuary integration. You can also find a complete example setup script in [Estuary's docs](#).

-  **Name:** a unique name of your choice for your integration.
-  **Data plane:** choose a cloud provider and region. It's often optimal to pick the same provider and region that you use for Snowflake.

Under the Endpoint Config section, set up:

-  **Host:** your Snowflake account URL (minus the protocol), such as: `orgname-accountname.snowflakecomputing.com`.
-  **Database and Schema:** specify the database to load data into. If you created resources using Estuary's example script, for example, you would use `ESTUARY_DB` and `ESTUARY_SCHEMA`.
-  **Warehouse and Role:** optionally set the warehouse and role as well. If these aren't specified, Estuary will use the default.



The screenshot shows the Estuary web interface for creating a materialization. The left sidebar contains navigation links: Welcome, Sources, Collections, Destinations (highlighted), and Admin. Below these are options to Toggle Color Mode and Collapse Navigation. The main content area is titled '2. Endpoint Config' and contains the following fields:

- Host (Account URL) ***: `orgname-accountname.snowflakecomputing.com`. Description: The Snowflake Host used for the connection. Must include the account identifier and end in `.snowflakecomputing.com`. Example: `orgname-accountname.snowflakecomputing.com` (do not include the protocol).
- Database ***: `ESTUARY_DB`. Description: The SQL database to connect to.
- Schema ***: `ESTUARY_SCHEMA`. Description: Database schema for bound collection tables (unless overridden within the binding resource configuration) as well as associated materialization metadata tables.
- Warehouse**: `ESTUARY_WH`. Description: The Snowflake virtual warehouse used to execute queries. Uses the default warehouse for the Snowflake user if left blank.
- Role**: `ESTUARY_ROLE`. Description: The user role used to perform actions.

You will also need to select a Snowflake **Timestamp Type**. Options (TIMESTAMP_LTZ, TIMESTAMP_NTZ, and TIMESTAMP_TZ) map directly to Snowflake's timestamp column data types. If you already have an explicit `TIMESTAMP_TYPE_MAPPING` set for your Snowflake account, you must choose this type in Estuary.

Under **Authentication**, make sure you're using the **Private Key (JWT)** option. Add your Snowflake user and your private key. You can either copy the key to the clipboard using `cat rsa_key.p8` or upload the p8 key with the file selector.

Set the Sync Schedule for predictable batches with Snowflake bulk loading, or configure tables to use Delta Updates to ingest data with Snowpipe Streaming.




Connection Security with PrivateLink

PrivateLink and similar cloud services offer enterprise-grade security and compliance, allowing you to connect services without exposing traffic to the public internet.

Estuary users with private or BYOC (Bring Your Own Cloud) deployments can connect their Estuary data plane with their Snowflake instance using AWS PrivateLink or Azure Private Link. On the Snowflake side, PrivateLink is a Business Critical Edition (or higher) feature.

Specific setup instructions depend on the cloud provider. Broadly, though, a PrivateLink implementation will consist of:

-  Setting up a load balancer

-  Routing traffic through the load balancer to your instances
-  Sharing resource name and zone information with Estuary
-  Using the Estuary-provided DNS name as the host name in your connector configuration

Learn more about PrivateLink connections in [Estuary's](#) and [Snowflake's](#) docs.

Connector Management with Infrastructure-as-Code

While Estuary's UI can make it simple to set up connectors and ramp up integrations quickly, manually updating configurations in the UI bypasses the review and deployment processes that most modern enterprises require.

Instead, you can work directly with Estuary's `flow.yaml` configuration files. You can save your configuration to a workspace repository with branch protections to ensure changes are reviewed and set up actions to automatically publish those changes once merged.

You can find detailed information on working with Estuary's `flow.yaml` configuration files [in this guide](#).

Whether you choose to make changes through the UI or through the CLI, Estuary provides a detailed change history for each of your connectors that allows you to compare configs, track when changes were made, and who made them.

Snowflake Setup with Fivetran

Fivetran is an ELT data platform focused on batch workflows, with some lower-latency options when using the more complex self-hosted HVR product.

Fivetran's Snowflake integration focuses on the bulk loading data ingestion method.

Fivetran Configuration

Fivetran uses the Snowflake resources mentioned in the “Common Setup Steps” section. Make sure these are created before you start configuring your Fivetran integration. You can also find a complete example setup script in [Fivetran's docs](#).

Setup steps may vary slightly depending on the deployment model you use. For this example, we'll assume a SaaS deployment.

To start setting up, add a new **destination** from the Destinations page in the Fivetran dashboard. Provide a name and choose **Snowflake** for the destination type.

You will then need to configure your connection, such as:

-  Selecting your desired deployment model (SaaS)
-  Provide and authenticate an **external storage stage** (AWS, GCS, or Azure)

- Provide Snowflake host information depending on whether you want to connect directly or through a service like PrivateLink
- Enter details for the Snowflake **user**, **database**, and user authentication such as a **private key**
- Optionally set your desired **role** and **warehouse**
- Select a **data processing location**
- Choose your desired **timezone**

Once your Snowflake destination is configured as desired, Save & Test the connection.

Comparing Snowflake Integration Solutions

While there are many Snowflake connectors, integrations, and variations out there, we've now covered some of the top methods to get data into your Snowflake instance.

From hands-on native integrations to streamlined third-party solutions with built-in robustness, there's a method to load data into Snowflake for any set of requirements.

This section will cross-compare the third-party platforms we've reviewed and provide some considerations on choosing the right solution for your needs, whether first- or third-party.

Quick Comparison Chart: Third-Party Integrations

	Airbyte	Estuary	Fivetran
Brief Description	Built on an open-source base, Airbyte is known for a wide range of connectors	The right-time data platform, supporting flexible data strategies for all types of data movement	An established ELT platform, Fivetran provides visual pipeline setup and transformations
Snowflake Integration Support	Full refresh or batch incremental source and destination connectors	CDC source capture and efficient destination connector	Full refresh or batch incremental source and destination connectors
Right-Time Flexibility	Low flexibility: Batch only	High flexibility: Batch, micro-batch, and real-time streaming	Medium flexibility: Batch-focused, with more frequent batch schedules based on customer tier
Ease of Use	Takes time to learn, set up, implement, and maintain	Low- and no-code pipelines with smart defaults that make it quick to get up and running	Easy to use pipeline tools for cloud, with more advanced setup for other options like HVR
Connectors	Ecosystem of 600+ connectors, counting marketplace integrations	200+ high-quality in-house connectors with new connectors created on request	<300 regular connectors with 450+ “Lite” API connectors

	Airbyte	Estuary	Fivetran
Deployment Options	Open source, public cloud	Open source, public cloud, private cloud, BYOC	Cloud, hybrid, self-hosted HVR
Reliability	Medium Customers often report needing a lot of maintenance and troubleshooting once pipelines are set up	High Designed with separate control and data planes, replicas, and the ability to pick back up where you left off	Medium-High While an established platform, outages or updates can occasionally affect pipelines
Support	Basic Premium support is still relatively new, with only limited forum support previously available	Top-tier Customers consistently mention great support experiences with fast turnaround and knowledgeable staff	Standard Customers occasionally mention slow support times or requiring multiple sessions to solve issues
Cost	Low vendor costs, with higher costs for engineering and troubleshooting; inefficient connectors can lead to high costs in connected systems	2-5x lower vendor costs than similar solutions; low engineering costs because "it just works"	High costs with unpredictable MAR-based pricing

Data Integration Guides: Where Does Your Data Go Next?



Data Integration Guides: Where Does Your Data Go Next?

Whether you load your data into Snowflake via one of their native data ingestion offerings or a third-party integration, Snowflake isn't necessarily the be-all, end-all for your data.

After all, one of the most powerful use cases for real-time analytics is that you can send new data to Snowflake, use Snowflake to crunch complex queries, and then send data back, all in time to inform an end user or augment a time-sensitive workflow.

As this guide is mainly about the initial data ingestion step, we won't linger too long here, instead simply covering a few common options to enhance your data or send it to the next stage.


Adding dbt to the Mix

One common method data teams use to get data ready for prime time after raw data has landed in Snowflake is to progressively clean datasets with dbt.

You can set up dbt projects to define expected data models, transform data, and create tables or views of the cleaned, transformed data without losing the originals.

This can make it easier to backtrack and verify if something about the results seems suspect.



You can add dbt to your Snowflake data in different ways:


-  Snowflake supports [dbt Core](#). You can define, execute, and schedule dbt projects with Snowflake objects.
-  Third-party integrations often have their own dbt integration options:
 - Fivetran supports [dbt Core](#) or [dbt Cloud](#) integrations
 - Estuary supports [dbt Cloud](#) integrations

Using Data in Cortex or Other AI Use Cases

One reason to move data into Snowflake is to, of course, actually make use of it in Snowflake. Snowflake provides an array of features to analyze your data and make it accessible, including Cortex.

Cortex is Snowflake's AI integration, focused specifically on LLMs. Cortex can be broken down into various sub-features that help you understand your unstructured data, including:

-  **Cortex Search:** Search across your textual data, either to enhance search results or power RAG applications
-  **Cortex Analyst:** Ask questions about your structured data to generate associated SQL queries

 **Cortex Agents:** Create agents to perform tasks based on results from Cortex Analyst and Cortex Search

You can view more about Snowflake's Cortex AI and ML features [in their docs](#).

Reverse ETL or Snowflake as a Source

Snowflake doesn't have to be your final destination. Once you aggregate all your data in one place, clean it up, and analyze it, you may want to send the results back out into the world.



Many third-party data integration platforms that load data into Snowflake can also take it out again. Here, as with ingestion, not all integrations are created equal, so you should review your options carefully. For example, loading data from Snowflake in a batch that reprocesses data again and again is much different than highly efficient CDC, which captures changes incrementally as they occur.

The third-party platform should also, of course, support your intended destination, whether that's a SaaS platform or your transaction database.

Your calculations should also take into consideration that Snowflake charges fees for data egress besides any compute costs for accessing the warehouse.

You can review supported features and other information about Snowflake source connectors for:

 [Airbyte](#) (Full refresh or incremental batch)

-  [Estuary](#) (CDC capture)
-  [Fivetran](#) (Full refresh or incremental batch)

Estuary



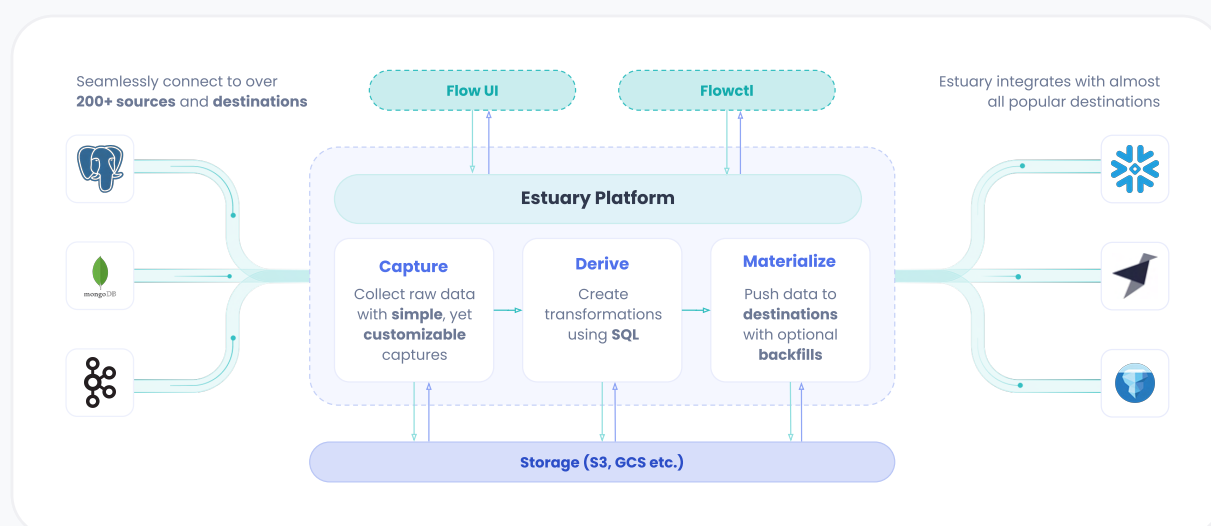
Who We Are

Estuary is the Right-Time Data Platform, allowing you to stream at millisecond latency or schedule predictable batches as your use case requires. With simple setup and automatic schema evolution, Estuary takes a lot of the guesswork out of data integration.

Estuary in Your Snowflake Stack



Estuary is a great way to ingest data into Snowflake. Offering hundreds of high-quality connectors for databases, SaaS platforms, and other sources, Estuary provides durable integration with smart defaults and flexible customization.

Estuary Architecture - Seamlessly store, manage, and connect data



Estuary Architecture

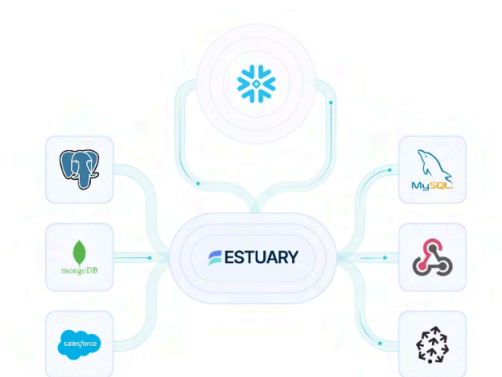
Estuary in Your Snowflake Stack

-  **Batch ingestion:** Estuary efficiently merges data into your warehouse using COPY INTO queries. Set your desired sync frequency.
-  **Streaming ingestion:** Use Snowpipe Streaming without the hassle to ingest data as soon as possible. Set “Delta Updates” mode to use Snowpipe Streaming.

[Learn How To Set Up A Snowflake Destination Connector](#)

Capturing Data From Snowflake

You can also capture data from Snowflake into Estuary for reverse ETL use cases. Estuary’s Snowflake capture connector uses CDC for efficient, low-latency data movement.



[Learn How To Set Up A Snowflake Source Connector Here](#)

Right-Time Integration with Estuary: A Strategic Advantage for Snowflake Customers

Right-time data

Ingest data into Snowflake using efficient bulk loading or ultra-fast Snowpipe Streaming. Easily update your cadence in the future as your requirements change.

Ease of use, with swift setup

Using smart defaults and automation for tedious tasks, pipeline setup is fast and seamless with Estuary's no-code connectors. Implement a PoC or complete solution in hours or days.

Hear from the customer: [Find out how Flash Pack set up real-time data capability in days.](#)

Automatic schema inference and evolution

Integrating with hundreds of tables is a breeze: Estuary automatically detects fields and their data types from your sources and can automatically keep schemas up to date as those fields change.

Flexible deployments

Use public or private cloud offerings, or even bring your own cloud. Keep data within certain regions for compliance and make use of VPCs with private connection options for maximum security.

Enterprise-grade reliability

Using a unique decoupled architecture, pipeline sources aren't tightly tied to any one particular destination, and the control plane remains separate from the data plane. Along with built-in replication, Estuary provides enhanced durability, helping to minimize and manage outages.

Hear from the customer: [Find out how David Energy prioritizes reliability by choosing Estuary.](#)

High-quality Snowflake connectors

Estuary focuses on best-in-class efficiency for all integrations, whether that means using CDC for the Snowflake source connector or finetuning merge queries for the Snowflake destination connector. Experience similar efficiency with all the data sources you want to load into Snowflake, whether database, SaaS, or message streaming.

Cost efficient

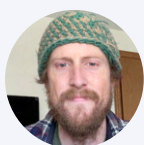
Estuary's vendor costs are some of the lowest in the business, often 2-5x lower than competitors with some customers reporting a 10x difference against comparable solutions. Annual contracts can reduce your costs further.

Hear from the customer: [Find out how Xometry cut costs by 60% while securing their data with a private deployment.](#)

What Our Customers Say

“

Estuary and Snowflake are like best friends. They speak very well together.



Evan Dixon
Data Engineer

The CURRI logo, featuring the word "CURRI" in a bold, black, sans-serif font inside a light gray rounded rectangle.

“

Estuary has been a game-changer for Headset's data infrastructure. Compared to our previous solutions, it has dramatically improved reliability while reducing our overall costs significantly. The real-time ingestion capabilities ensure that our analytics are always powered by the freshest, most accurate data without the operational headaches we faced before.



Scott Vickers
CTO

The HEADSET logo, featuring a colorful geometric icon followed by the word "HEADSET" in a bold, black, sans-serif font, all inside a light gray rounded rectangle.

What Our Customers Say

“

We needed something self-serve, fast, and reliable, and Estuary delivered exactly that. It's a huge unlock for our operations, reporting, and machine learning.



Uri Vinetz

Director of Data

The logo for livble, featuring the word "livble" in a green, lowercase, sans-serif font.

“

50% cheaper and much faster. Estuary unlocked data that was cost prohibitive.



Brandon Besash

Director of Business
Intelligence

The logo for Glossier, featuring the word "Glossier." in a bold, black, sans-serif font.

What Our Customers Say

“

Estuary has been a pleasure to work with and has significantly modernized our data infrastructure, delivering real-time and scalable processes that will significantly impact company-wide operations. Every data-driven organization should be looking at Estuary today.



Andrew Woelfel

Senior Manager,
Data Engineering and Analytics



Appendix: Helpful Resources

Right-Time Data

Read the [Right-Time Data Manifesto](#)

Snowflake

Docs

Learn about [Snowflake's low-level data ingestion options](#)

Find out about [data integration with Openflow](#)

Other resources

Read more about [Snowpipe Streaming](#) for real-time data

Integration Platforms

Airbyte

Learn about Airbyte's [source](#) and [destination](#) connectors for Snowflake

Estuary

Learn about Estuary's [capture](#) and [materialization](#) connectors for Snowflake

Watch a step-by-step [guide for Snowflake connector setup](#) with Estuary

Discover user insights with [Estuary's success stories](#)

[Schedule a call](#) to learn more or [try the platform for free](#)

Fivetran

Learn about Fivetran's [source](#) and [destination](#) connectors for Snowflake