snowflake®

DEVOPS FOR DATA APPS ON SNOWFLAKE

CHAMPION GUIDES

TABLE OF CONTENTS

INTRODUCTION

The benefits of building a DevOps culture in software companies are clear. DevOps practices integrate once-siloed teams across the software development life cycle, from Dev to QA to Ops, resulting in both faster innovation and improved product quality. As a result, most software development teams have deployed tools to enable DevOps practices across their workflow. Source control tools, such as Git, enable multiple developers to work on the same software simultaneously. Each developer checks out the code and develops features in a private branch in their own development environment. And continuous integration and continuous delivery (CI/CD) tools automate the process of running unit tests on the new code, integrating it into new releases, pushing it to QA and staging for end-to-end testing, and, in some cases, deploying it to production. The benefits are so compelling that most companies that build software have a strong DevOps culture and already have a mature tool chain in place to enable it.

But for developers that need to embed a data platform into their applications to support data intensive workloads, significant challenges emerge. For starters, DevOps for databases is much more complex than DevOps for code because databases contain valuable data, whereas code is stateless. Moving code to and from source control, or pushing it to stages and rolling it back is simple, cheap, and fast. Doing the same with large data sets is costly and time consuming. In addition, modifying database schemas can break the code or even result in data losses. In this environment, how do you synchronize changes in the application with changes in the database? How will your existing tools and workflows adapt to support the data platform? What new tools will you need?

This ebook describes how the Snowflake Data Cloud enables application developers to simplify their DevOps processes, the key product capabilities that help with DevOps, and the ecosystem of DevOps tools that work with Snowflake.

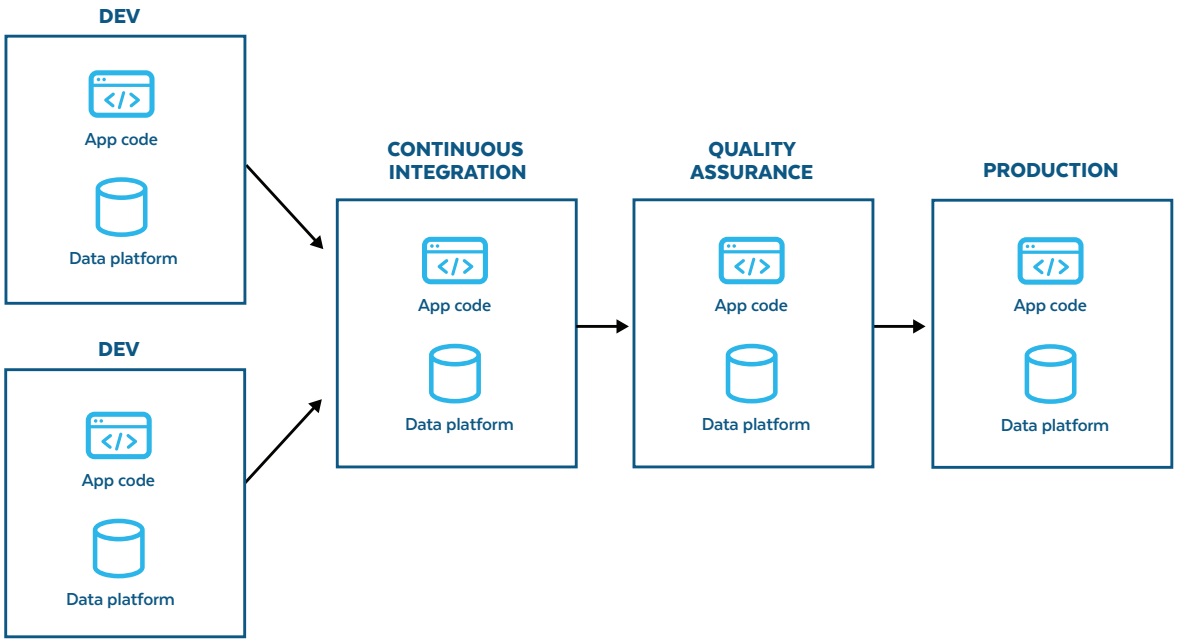


Figure 1: Data app DevOps pipeline

HOW SNOWFLAKE ENABLES DATA APP DEVELOPERS TO SIMPLIFY DEVOPS PROCESSES

Snowflake is designed to enable developers to build data-intensive applications with no limitations on performance, concurrency, or scale. Because of its multi-cluster, shared data architecture, it scales horizontally and vertically on demand, delivering fast response times regardless of load. It supports structured, semi-structured, and unstructured data for deeper insights. Native support for JSON and standard SQL simplifies data pipelines and helps deliver fresh insights from all data. And because it is delivered as a service, Snowflake improves developer productivity by removing the need to maintain infrastructure.

Here are 10 ways in which Snowflake simplifies DevOps processes for application developers:

1. INSTANTLY CREATE ANY NUMBER OF ISOLATED ENVIRONMENTS

Snowflake enables developers to stand up as many isolated, ACID-compliant, SQL-based compute environments as needed. There is no need to procure, create, and manage separate cloud resources for each stage of the pipeline. Developers can create new environments in seconds with standard SQL queries, auto-suspend idle environments, and delete environments when the testing has been completed. Additionally, developers can use standard SQL, rather than proprietary APIs, to perform these operations.

The core of Snowflake's unique architecture is the ability to run any number of isolated workloads that all access the same data. Compute environments in Snowflake, called virtual warehouses, are completely independent. This enables the creation of isolated compute environments for each stage in the pipeline, so that no matter how intense the processing is in one environment, the others are not affected.

2. REDUCE SCHEMA CHANGE FREQUENCY WITH VARIANT DATA TYPE

Snowflake supports semi-structured data natively using a proprietary VARIANT data type. A VARIANT column is typically used to ingest JSON or other semi-structured data directly into a relational table without defining the schema ahead of time. Once loaded, JSON data can be queried using SQL or even joined with other structured data. This capability simplifies data pipelines significantly as there is no need to modify and retest code every time new parameters are added. (See Figure 2.)

New features commonly require schema changes. But schema changes can be expensive operationally as they require developers to coordinate their code changes with updates to the database schema. With Snowflake, rather than add new columns and change schema frequently, many developers choose to store their data in JSON and apply a schema at query time (schema-on-read). Customers often also choose to store data that might originally be structured (like from a traditional RDBMS) in a semi-structured JSON format to avoid managing schema changes in a traditional fashion. Using VARIANT columns in this way significantly reduces the DevOps operational burden for app developers.

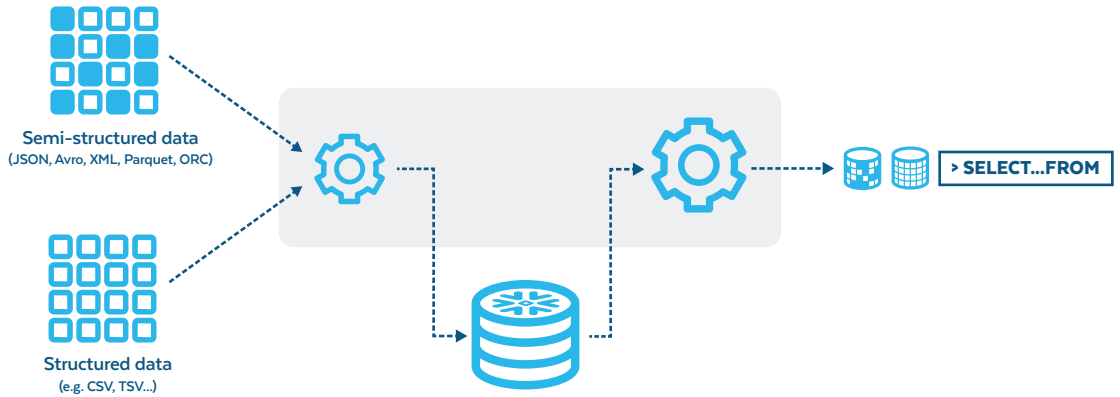


Figure 2: Native support for semi-structured data

3. RAPIDLY SEED PREPRODUCTION ENVIRONMENTS WITH PRODUCTION DATA

Seeding preproduction environments with production data presents one of the biggest DevOps challenges for application builders. As most developers know, it's risky to validate feature changes using test data. Even if the schema is the same between production and preproduction, production data is still important to properly validate code changes.

With traditional databases, seeding preproduction environments with production data is so costly and time consuming that it can result in schedule delays or compromised product quality. With Snowflake, seeding preproduction environments can happen in seconds.

Snowflake offers two mechanisms for seeding a preproduction environment with production data. Snowflake Secure Data Sharing is used when the environments are on separate Snowflake accounts, and Zero-Copy Cloning is used when the environments are on the same account. (See Figure 3.)

Zero-Copy Cloning creates a copy of live data instantly in metadata, without the need to duplicate or move data, saving storage costs and time. Only modified data requires additional storage. For example, to create a development database from production data in Snowflake, all you need to do is run a **CREATE DATABASE Dev CLONE Prod;**.

That's it! A single command clones the entire database almost instantly and without incurring additional storage costs. You can now develop in a safe environment. For a more complete example, including how you can use zero-copy clones to promote the changes from Dev to Test to Prod, see Saving Time & Space: Simplifying Devops with Fast Cloning.

	WHEN USED	HOW IT WORKS	COPIES DATA
ZERO-COPYING CLONING	Environments are on the same Snowflake account.	Metadata operation with pointers to the same data. No data movement.	No
SECURE DATA SHARING	Environments are on separate Snowflake accounts.	Read-only database is shared from the production account to preproduction environments.	As needed

Figure 3: Methods to seed environments with production data

Snowflake Secure Data Sharing enables access to live data from a provider account to one or many consumer accounts and is typically used to share data with partners or with other departments. Sharing happens through Snowflake's services layer and metadata store without copying or moving data, so no additional storage charges are incurred and all consumers have access to the same data set. Using Secure Data Sharing in a DevOps setting allows developers to avoid the lengthy backup and restore operations normally associated with seeding an environment with production data. Data can be copied with a **CREATE TABLE AS SELECT (CTAS)** operation if needed, significantly reducing the time required.

4. EASILY ROLL BACK WITH TIME TRAVEL

Handling errors with the automated release of a database and related application code is a significant challenge. With traditional systems, developers have to make backups before releasing new changes and then restore data in the event of a failure. But both backup and restore operations are very time consuming and costly—especially for large amounts of data.

Snowflake Time Travel simplifies almost all of the effort required to handle errors and rollbacks in a CI/CD process. With Time Travel, objects such as tables, schemas, and databases that have been changed or deleted can be easily restored or accessed programmatically at a point in time within the previous 90 days without accessing costly backups (see Figure 4). Developers can easily create scripts to roll back in the event of a failure with a simple **CLONE AT** or **BEFORE** command (followed by an **ALTER TABLE ... SWAP** command).

5. INSTANTLY SCALE ENVIRONMENTS TO RUN JOBS FAST AND COST EFFECTIVELY

With Snowflake's per-second consumption pricing model, organizations pay only for the time to run the job, no matter the cluster size. This means preproduction environments can be built and scaled to whatever size is required—and developers no longer have to suffer through slow dev cycles caused by environment size restrictions. The same benefit applies to production workloads. DevOps teams can scale the production environment to run a big process in a fraction of the time and scale it down when the process ends. For most jobs, the price will be the same.

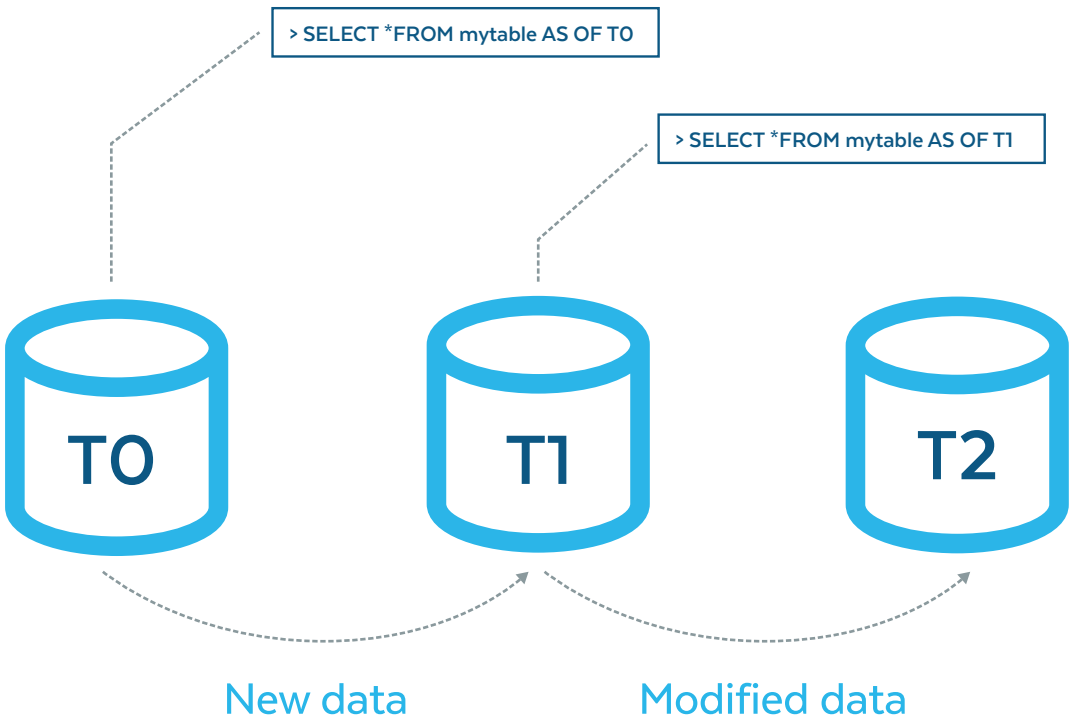


Figure 4: Time Travel live backups

6. INCREASE VELOCITY WITH STANDARD SQL

Unlike alternatives that often require the use of proprietary query languages, Snowflake supports ANSI SQL for all data: structured, semi-structured, and unstructured. As a result, Snowflake customers benefit from the rich ecosystem of SQL tools available for BI, Data Engineering, and DevOps. Snowflake is an ACID-compliant SQL implementation that enables querying all data with SQL and updating it with multi-statement transaction support. Additionally, all administrative tasks such as provisioning new warehouses can be done through SQL. There is no need to administer the platform through proprietary APIs.

Because engineers already know SQL, they make fewer mistakes. This results in faster sprints and more streamlined CI/CD workflows, with fewer rollbacks and higher overall quality. Mistakes are caught earlier in code reviews because it's easier to determine how a query will behave when everyone is familiar with the language.

7. USE YOUR PREFERRED PROGRAMMING LANGUAGE

In addition to standard SQL, Snowflake supports the most popular programming languages including Python, Node.js, Go, .NET, and Java, so developers can code in their preferred language and leverage their existing tools to control and manage their Snowflake code. All SQL commands can be wrapped in any of the supported programming languages. Snowflake also has drivers for JDBC and ODBC, expanding the ecosystem of compatible tools.

Broad language support also enables developers to take advantage of serverless compute options available from their cloud provider to easily integrate with Snowflake. Developers can use serverless functions to perform tasks, including administration, data processing, and consumption directly from their cloud platform and preferred tools.

Snowpark offers developers even more flexibility: they can now write Java, Python (currently in public preview), and Scala code and run it directly in Snowflake without needing to maintain a separate compute service such as Spark, container, or serverless functions.

The Snowpark developer experience is all about extensibility. It's a powerful option for teams adopting DevOps principles for building data pipelines, as it allows for change control; automation of building, testing, and deploying code; and promotion of best practices for modularity and reusability.

8. REDUCE DEVOPS BURDEN WITH NEAR-ZERO MAINTENANCE

Snowflake automates many of the tasks that traditionally burden DevOps so they don't have to manage infrastructure, install patches, or perform backups. There are no virtual machines or networking gear to provision, configure, and manage. Snowflake rolls out software updates automatically so all environments are always running on the latest version. In addition, Snowflake automates most optimization tasks and eliminates the need for performance tuning, indexing, vacuuming, and partitioning.

High availability, governance, and data security are also built-in. Snowflake customers can replicate databases and keep them synchronized across multiple accounts in different regions and different cloud providers, ensuring data durability and availability. In a massive outage, customers can

immediately fail over to an available region or cloud provider to continue operations, regardless of the amount of data. Security is also built-in, including encryption of data in transit and at rest, granular role-based access control, and optional multi-factor authentication.

9. SIMPLIFY DATA PIPELINES WITH STREAMS AND TASKS

Snowflake Streams and Tasks are built-in capabilities to orchestrate integrating and transforming data in Snowflake. Streams process only the data that has changed in the source table. Tasks automate the necessary transformations and loading on a predefined schedule. Multiple tasks can be chained to implement a complex data pipeline. The Streams and Tasks feature helps simplify workflows because they are built into Snowflake and remove the need to write custom code or rely on a third-party data integration tool, which would add operational complexity.

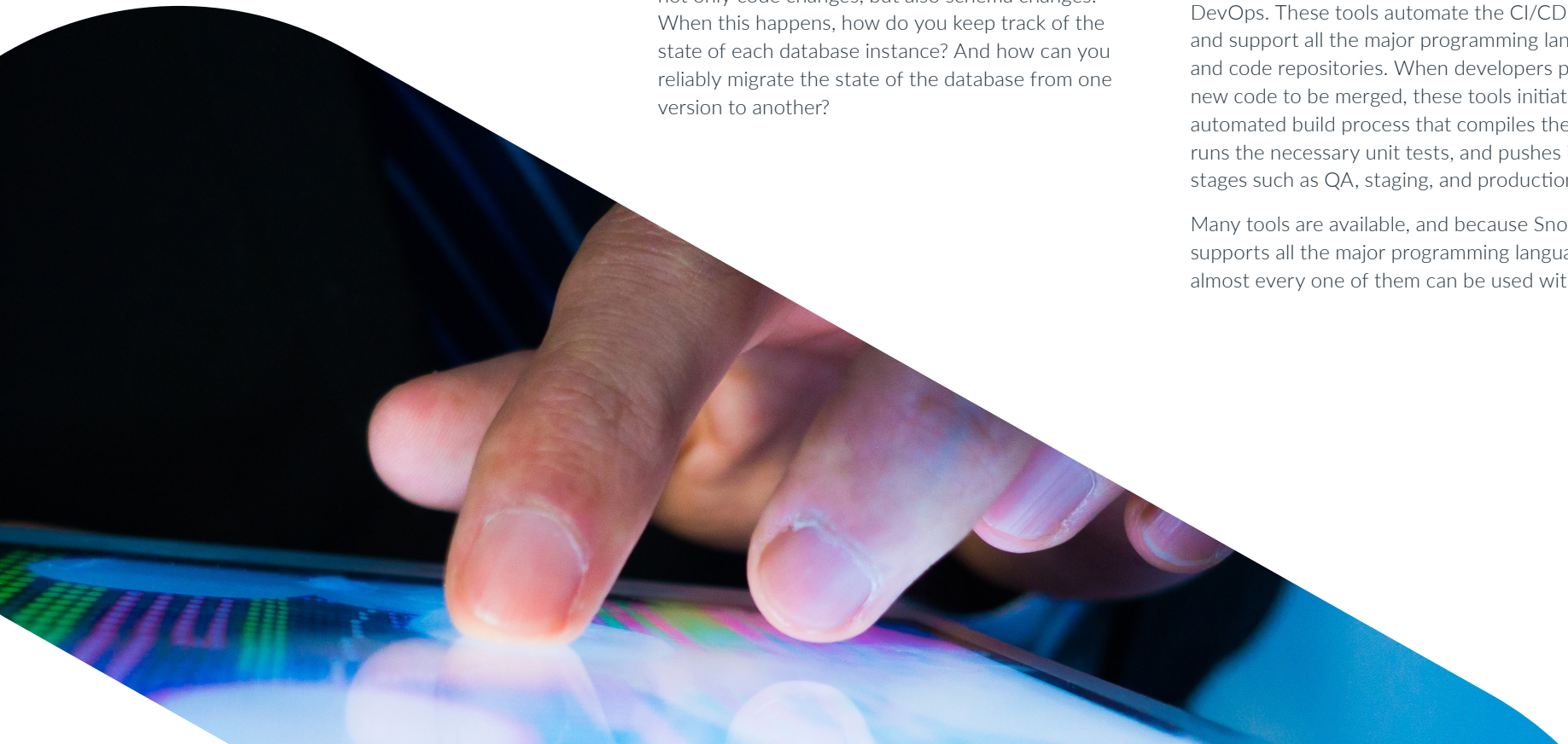
10. SAVE TIME WITH REAL-TIME INTEGRATION WITH EXTERNAL SERVICES

Snowflake External Functions simplifies integration with external services. Developers can use External Functions to call third-party or custom services that are stored and executed outside of Snowflake. These remote services can be written using any HTTP server stack, including cloud serverless compute services such as AWS Lambda.

In addition, the External Functions feature helps streamline DevOps processes by reducing the number of tools and steps. Application developers often need to integrate third-party libraries and APIs into their apps, for example, for geocoding addresses, ML scoring, data-masking services, or advanced custom business logic. With External Functions, developers can apply those services to data within Snowflake in real time and do not have to augment the data statically before loading it to Snowflake, thus reducing the number of tools and steps and simplifying their DevOps workflow.

TAP INTO A BROAD PARTNER ECOSYSTEM

Many of Snowflake's built-in capabilities help simplify DevOps processes for developers building data applications. To extend those benefits and enhance flexibility, Snowflake has an ecosystem of partners that provide tools to implement end-to-end DevOps processes, including database change management, software automation, and data integration.



SET UP VERSION CONTROL WITH COMPATIBLE DATABASE CHANGE MANAGEMENT TOOLS

Most developers are familiar with using version control for their application code. Version control tools enable multiple developers to modify the same software without conflict, create reproducible builds, and roll back to earlier versions if necessary. But as developers embed a data platform as part of their applications, feature changes typically require not only code changes, but also schema changes. When this happens, how do you keep track of the state of each database instance? And how can you reliably migrate the state of the database from one version to another?

Database change management tools help developers do just that. An increasing number of database change management tools support Snowflake. These include Flyway, schemachange, Terraform, Liquibase, and Sqitch.

AUTOMATE CI/CD PIPELINES WITH YOUR PREFERRED SOFTWARE AUTOMATION TOOLS

Software automation tools are the lifeblood of DevOps. These tools automate the CI/CD pipelines and support all the major programming languages and code repositories. When developers push new code to be merged, these tools initiate an automated build process that compiles the code, runs the necessary unit tests, and pushes it to stages such as QA, staging, and production.

Many tools are available, and because Snowflake supports all the major programming languages, almost every one of them can be used with

Snowflake. Some of the tools specialize in continuous integration (CI), some in continuous delivery/deployment (CD), and some do both. Available CI/CD tools include GitLab, CircleCI, GitHub Actions, Jenkins, and Azure DevOps.

CHECK IT OUT

Get a demo of CI/CD with Snowflake using GitHub Actions, Terraform, and Schemachange. **Watch now »**

BUILD DATA PIPELINES WITH LEADING DATA INTEGRATION TOOLS

Data engineers use data integration tools, also called ETL/ELT tools, to manage loading and transforming data. In many cases developers need to clean, integrate, or model data in Snowflake before it can be used by the application. This can happen before or after loading the data.

There are three common options, depending on the specific requirements for each application:

- Leverage an ETL/ELT tool from one of several Snowflake Data Integration Partners
- Use Snowflake's native Task capabilities for transforming data after it's in Snowflake
- Create a custom process and run it, typically from a serverless compute service such as AWS Lambda or Azure Functions

For each option, developers manage the data pipeline code in source control and create a script to deploy the pipeline so it can be easily integrated into the overall DevOps workflow.

POWERED BY SNOWFLAKE PARTNERS AND DEVOPS

The Snowflake ecosystem includes hundreds of Powered by Snowflake partners, several of which have DevOps interests. Check out these two to start, and watch our YouTube series for more deep-dive interviews.

Software.com

Take a look at Software.com's DevOps Metrics Platform, which processes and analyzes data from a wide variety of sources to create insights into potential bottlenecks in the DevOps processes.

Watch now »

Copado

See a demo from Copado, an end-to-end, native DevOps platform for Salesforce built on Snowflake.

Watch now »

CONCLUSION

Snowflake enables developers to build massive-scale data applications with minimal DevOps burden. Although most app developers have mature DevOps workflows already in place, as they integrate a data platform into their application to support data-intensive workloads, significant new challenges emerge to synchronize changes in code with changes in the data platform. Managing CI/CD pipelines for databases is more complex because, unlike code, databases are stateful and contain valuable data.

Snowflake significantly simplifies DevOps workflows for app builders with features and capabilities such as Zero-Copy Cloning, Time Travel, native JSON support, near-zero maintenance, and support for standard SQL and all major programming languages. Developers can leverage their existing tools for version control and CI/CD automation, use Snowpark to build in their preferred language and run from a single platform, and take advantage of a broad ecosystem of tools for database change management and data integration.

Want more details about DevOps for data applications on Snowflake? Watch our webinar for key tools and recommendations for building on Snowflake.

ABOUT SNOWFLAKE

Snowflake enables every organization to mobilize their data with Snowflake's Data Cloud. Customers use the Data Cloud to unite siloed data, discover and securely share data, and execute diverse analytic workloads. Wherever data or users live, Snowflake delivers a single data experience that spans multiple clouds and geographies. Thousands of customers across many industries, including 506 of the 2021 Forbes Global 2000 (G2K) as of April 30, 2022, use Snowflake Data Cloud to power their businesses. Learn more at **snowflake.com**.