Powering Observability with Snowflake

This document explores the metrics, logs, and traces available to enable data observability for DevOps teams. It also describes partner integrations and other features that enable observability and troubleshooting.

By Adrian Lee

Contents

Overview

With the rise of application modernization, DevOps teams are increasingly adopting observability. The primary mandate of observability is to generate actionable insights of a system's internal state by analyzing various data points such as metrics, events, logs, and traces. By aggregating these metrics along various dimensions, DevOps teams are able to

better understand and manage their systems. In turn, DevOps teams will then be able to react to issues faster before they have a downstream impact on other systems.

So is it possible to enable data observability to help DevOps teams operate more efficiently and manage their data platforms? For example, how do we proactively look out for key metrics such as data freshness frequency, number of unauthorized attempts, cost control, number of times queries are failing, and much more? This enables DevOps teams to not only have timely intervention but perform more efficient troubleshooting of issues.

Fortunately, we are able to achieve this with the Snowflake Data Cloud. In this paper, we will explore the metrics, logs, and traces we have in Snowflake to enable data observability for DevOps teams. We will also explore other Snowflake features that enable data observability, how Snowflake assists in troubleshooting, and partner integrations.

Key metrics and logs provided by Snowflake

Data observability is critical for DevOps teams. It is therefore important that DevOps engineers are able to extract key metrics about their systems. Snowflake has a few different tables and views where you can easily find extensive and important logs and metrics. We will be looking at a few of them as noted in Table 1.

Table 1: Schema Details

Schema	**Details**
Information Schema	The Information Schema is a read-only schema that is created automatically for each database. This contains system views for all objects in that particular database as well as views of account-level objects (for example, non-database objects like roles and warehouses). The Information Schema also has table functions that return account-level usage data such as the **COPY_HISTORY, LOAD_HISTORY, TASK_HISTORY** or **QUERY_HISTORY**.
Account Usage Schema	The Account Usage Schema is a read-only schema as well as part of a shared system database called **SNOWFLAKE**. It contains account-related usage information such as **ACCESS_HISTORY** and **COPY_HISTORY**.
Organization Usage Schema	The Organization Usage Schema is a read-only schema and is part of a shared system database called **SNOWFLAKE**. It contains account-related usage information such as

	WAREHOUSE_METERING_HISTORY and **ORGANIZATION_USAGE**.
Data Sharing Usage Schema	The Data Sharing Usage Schema is a read-only schema and is part of a shared system database called **SNOWFLAKE**. It contains information about data exchange listings such as **LISTING_TELEMETRY_DAILY** and **LISTING_CONSUMPTION_DAILY**.

As seen in Table 1, Snowflake comes with two types of internal system-defined tables and views: Information Schema and Account Usage Schema. While these two types of system-defined tables and views use similar structures and naming conventions, there are some differences (see Table 2). A more detailed explanation of these differences can be found in the documentation at:
https://docs.snowflake.com/en/sql-reference/account-usage.html

Table 2: Comparison of Information Schema and Account Usage Schema

	Account Usage Schema	**Information Schema**
Includes dropped objects	Yes	No
Latency of data	From 45 minutes to 3 hours (varies by view)	None
Retention of historical data	1 year	From 7 days to 6 months (varies by view/table function)

Key metrics to be used for observability

An important aspect of observability is the ability to react proactively rather than reactively. As such, let us explore some key observability metrics that are important in diagnosing issues and improving performance.

Query history analysis
Having a history of all your queries is especially important when it comes to debugging or troubleshooting issues. For this, we are able to use the **account_usage.query_history** to extract logs consisting of important fields such as **EXECUTION_STATUS, USER_NAME, QUERY_TEXT, ROWS_INSERTED, ROWS_UPDATED** and **ROWS_DELETED**.

Query performance analysis

Understanding the performance of queries is another important metric. Using the **TOTAL_ELAPSED_TIME** and **EXECUTION_TIME** fields that can be extracted from the **account_usage.query_history,** we can actively monitor query run time and get alerts if the **TOTAL_ELAPSED_TIME** is taking longer than expected. This will enable us to analyze which users are responsible for taking up the request execution time.

Proactively spotting misconfigurations in cloud services
Using Snowflake, we are able to provide insight into failed authentication attempts so that teams can make timely interventions. The cloud services layer of Snowflake is in charge of authentication and other functions such as metadata store, which enables capabilities such as Time Travel and Zero-Copy Cloning. Because the cloud service layer is in charge of authentication, the database's **information_schema.login_history** can be queried to retrieve logs including data such as the **ERROR_MESSAGE, CLIENT_IP** and **EVENT_TIMESTAMP**.

Diagnosing ETL/ELT pipelines
ETL/ELT pipelines play an extremely critical aspect in data engineering services. It is of utmost importance that these pipelines are executed properly with few or no problems. Therefore, it is important that we can proactively monitor these ETL/ELT jobs. One way to monitor these ETL/ELT job metrics is with virtual warehouse performance metrics such as the throughput and latency. Using the **account_usage.query_history,** we are able to retrieve important metrics such as the **TOTAL_ELAPSED_TIME**, **WAREHOUSE_NAME, QUEUED_PROVISONING_TIME, QUEUED_REPAIR_TIME, QUEUED_OVERLOAD_TIME** and **QUERY_TYPE.**

Data freshness
An important aspect of data observability is data freshness, including the data's currency and the rate at which data is being changed over time for a particular database. We are able to observe and monitor data freshness by extracting the **TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, BYTES,** and **ROW_COUNT** of our tables from the **information_schema.tables** of a particular database from Snowflake.

Query references for metrics

Now that we've explored what kind of data observability metrics can be obtained from Snowflake, it's time to look at what's needed to support these metrics. Here are some sample reference materials and queries to help you get started.

- Sample account usage metrics
- Sample warehouse performance metrics

- Sample queries to analyze performance

- *Towards Data Science* blog post on data observability metrics

Empowering proactive actions with Snowflake

Along with collecting metrics and logs, it is important for DevOps teams to be able to aggregate, summarize, and provide timely mitigation. This helps the DevOps teams rectify issues before they become systemic risks.

Snowflake has a wealth of data that can help DevOps teams with their notification and mitigation needs, including:

- External functions
- Snowflake connectors
- Snowpipe error notifications
- Resource monitors and credit usage monitoring

We will explore each of these items in the following sections.

External Functions

External Functions call code that is executed outside of Snowflake. The remote execution code is otherwise known as a remote service. With external functions, Snowflake can integrate with an API gateway and serverless code to invoke mitigation methodology when an issue is detected. For instance, in the event DevOps teams get service-level objective (SLO) warnings and errors, they can use External Functions to integrate with PagerDuty—a SaaS incident response platform—to provide quick SLO response

Access External Functions documentation here.

Snowflake drivers and connectors

Snowflake has a variety of connectors and drivers such as the Snowflake Connector for Python, Snowflake Connector for Spark, Node.js driver, and ODBC/JDBC driver. In addition, Snowflake provides a SQL REST API which can be used to access and update data in Snowflake.

With these connectors, drivers, and SQL REST API, DevOps teams can build bespoke applications and integrations that will help manage Snowflake deployments and take proactive actions. For example, DevOps teams that are familiar with Python can develop a

Python Web Service with the Snowflake Connector for Python that deploys up-to-date network listing on Snowflake when a malicious IP is detected.

Access SQL REST API documentation here.

Find documentation for Snowflake Connectors and drivers here.

Snowpipe error notification

Snowpipe is Snowflake's continuous data ingestion service. It helps to load data within minutes after files are added to a stage and submitted for ingestion, and is meant to work with continuous data streaming use cases.

Snowflake has error notifications for Snowpipe through the use of Amazon Simple Notification Service (SNS). This feature is especially useful when we have streaming data workloads (for example, clickstream data going through Kafka to Snowflake via Snowpipe) and want to be alerted of potential errors when Snowpipe is being used to ingest the files. DevOps teams are also able to leverage Snowpipe REST API endpoints, such as insertReport, to fetch load reports' errors.

Learn more about Snowpipe error notifications here.

Get Snowpipe REST API Endpoint documentation here.

Resource monitors and credit usage forecasting

Another aspect of observability is optimizing your spend so you can more wisely plan your budget. Snowflake's resource monitors allow DevOps teams to control spend. By attaching available resource monitors to warehouses or to an account, DevOps teams are able to proactively monitor credit usage. Snowflake also enables teams to set actions on these resource monitors so they perform certain actions when a threshold is reached. For example, resource monitors support actions such as Notify and Suspend, which send a notification to all account administrators and suspend the assigned warehouse after the statement being executed has completed. A sample screenshot of a resource monitor is shown in Figure 1.

Access Resource monitor documentation here.

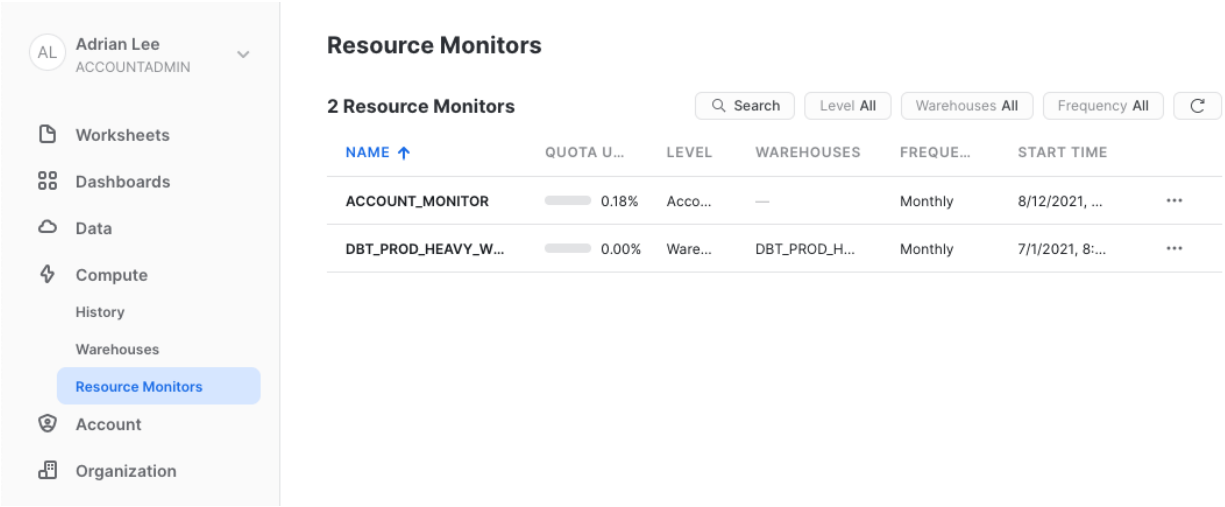

Figure 1: Resource Monitors

DevOps engineers are easily able to track credit consumption at a warehouse level across a period of time. When combined with resource monitors, this capability enables teams to perform capacity planning and credit forecasting, as shown in Figure 2.

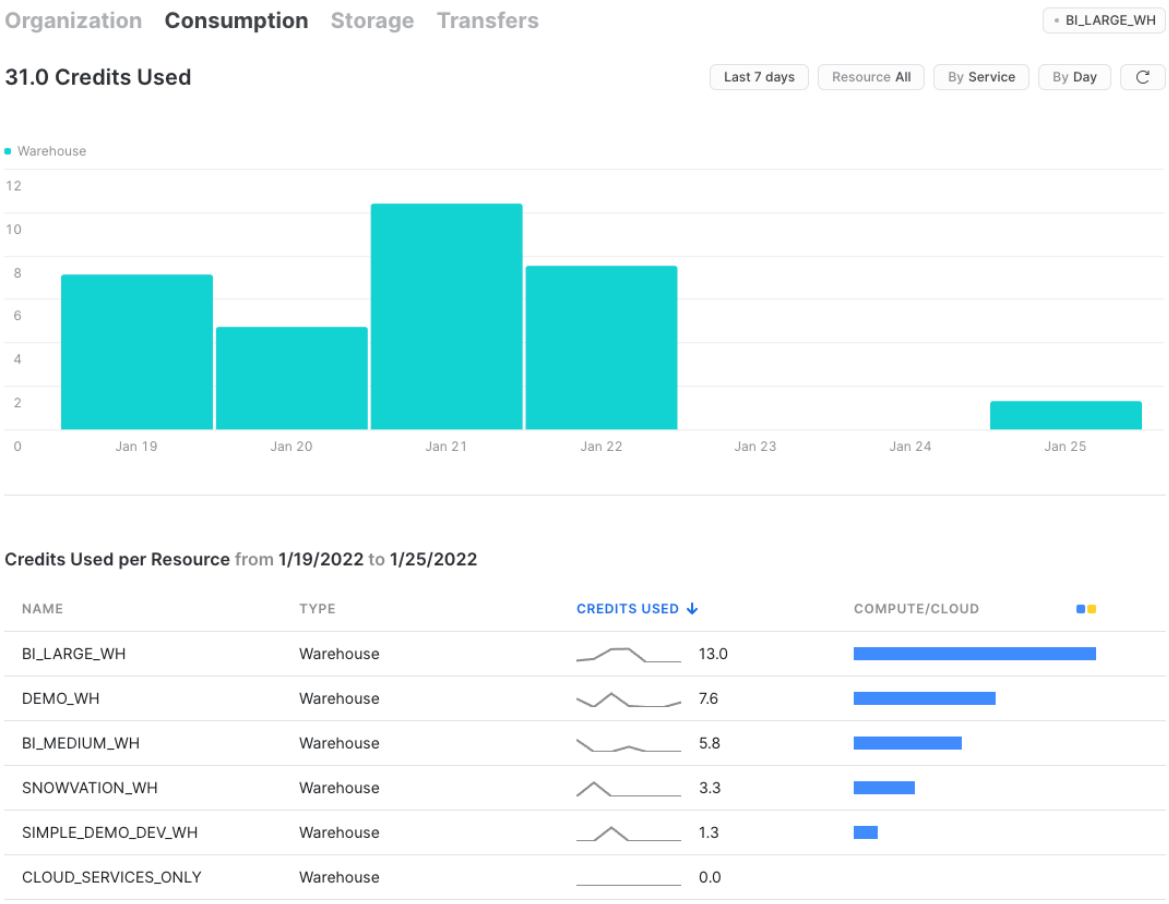

Figure 2: Consumption usage tracking

All of these Snowflake capabilities help enable proactive notification and mitigation. Please note: This list of functions is current as of March 2022 and may change, as Snowflake is constantly innovating to add new capabilities to the platform.

Troubleshooting with enhanced visibility

While taking proactive measures is essential, it is equally important for DevOps teams to be able to pinpoint the specific root causes of an identified problem. This can be achieved with enhanced visibility that helps DevOps teams speed up the troubleshooting process. In this section, we will look at how Snowflake provides enhanced visibility through the use of:

- Snowsight, which is Snowflake's web application
- Third-party business intelligence (BI) tools such Tableau and Microsoft Power BI

Snowsight

Snowsight is Snowflake's web application with integrated querying, dashboarding, monitoring, and administration; you can access it through the "Snowsight" icon in the old console. Snowsight helps you easily build your visualization dashboards by simply running SQL statements. DevOps teams can leverage this capability to build custom SQL-driven dashboards for troubleshooting based on the metrics discussed in the previous section on key metrics and logs. These dashboards (see Figure 3) can be directly shared in Snowflake across teams within the organization.

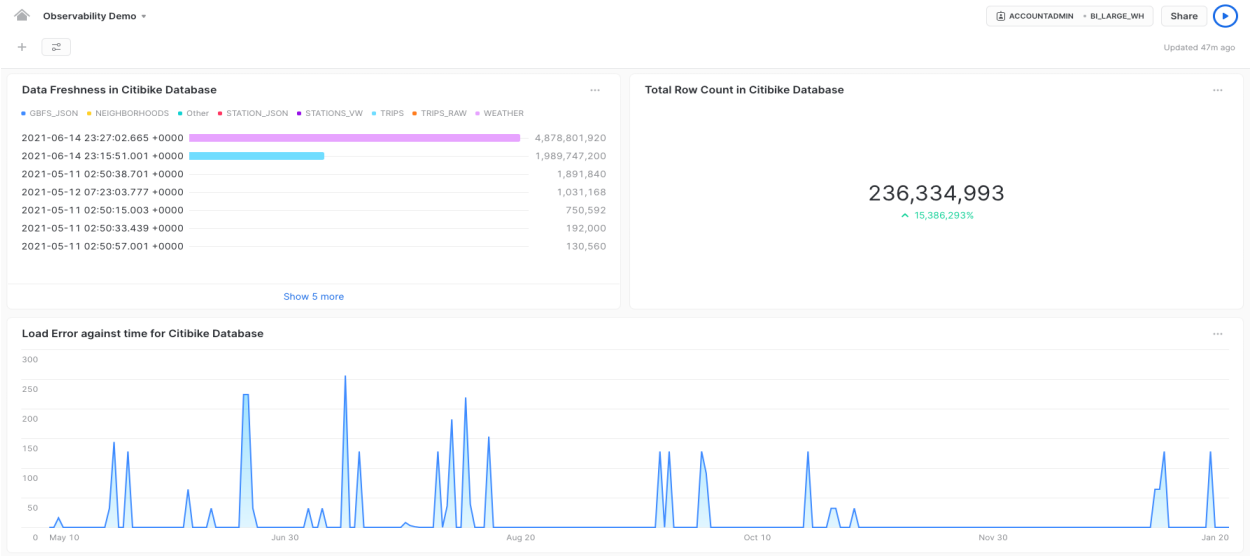

Figure 3: Example of a Snowsight observability dashboard

Troubleshooting breached user access

In the case of security breaches, such as a breached user, DevOps teams must find the breached user quickly so that they can analyze what other role access may have been compromised. Snowsight helps DevOps engineers track access control permissions graphically to analyze if there are any security breaches (see Figure 4).

Figure 4: Role access tracking in graph format

Troubleshooting failed data loading

When data loading fails, troubleshooting helps us better understand why it failed, as this could indicate a much more deeply rooted issue. Snowflake enables the copy history of a particular table to be viewed in a visual manner. This in turn enables DevOps engineers to trace the copy history (when the data was loaded, the size of the data, rows of data loaded, and status of the load) to see if there were any failures in copying data. A screenshot of this Copy History capability can be seen in Figure 5.

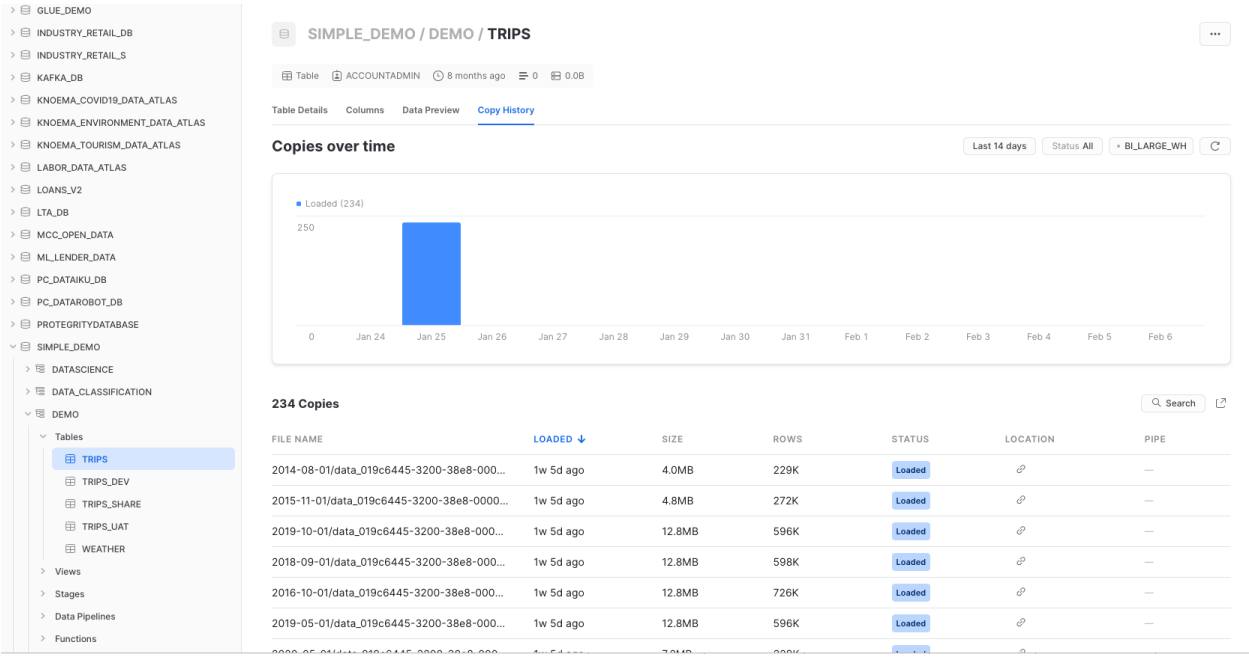

Figure 5: Copy History

Access documentation on Snowflake's new web interface here.

Tableau, Microsoft Power BI, and Sigma

Tableau, Microsoft Power BI, and Sigma have readily available dashboards that you can use to visualize your metrics. Here are some sample reference materials and blogs with step-by-step guidance to help you get started:

1) Blog by Tableau to monitor and understand Snowflake account usage: https://www.tableau.com/about/blog/2019/5/monitor-understand-snowflake-account-usage
2) Blog post on using Power BI to understand Snowflake account usage: https://medium.com/analytics-vidhya/snowflake-power-bi-snowflake-usage-report-f628dadbdc85
3) Blog post on dashboard templates provided by Sigma for Snowflake account usage: https://help.sigmacomputing.com/hc/en-us/articles/360045983174-Snowflake-Usage-Templates

Partner integrations

We previously discussed what kind of metrics, logs, and traces can be obtained from Snowflake as well as what kind of actions we can integrate to introduce timely intervention. We have also explored how we can use various dashboards to manage our observability data.

However, there are a number of customers who are already using other tools for observability and want to see Snowflake integrated readily into those products. Let's take a look at third-party partner observability platforms that have ready connections to Snowflake, and how some Snowflake's partners have built out their own observability services on top of Snowflake.

Available partner observability integrations

Grafana is a multi-platform open source analytics and visualization observability platform that has various plugins with databases and data platforms. Figure 6 shows Grafana integrated with Snowflake. It also comes with action trigger scripts such that when a threshold for a certain metric is crossed, an alert is triggered and custom scripts can be executed.

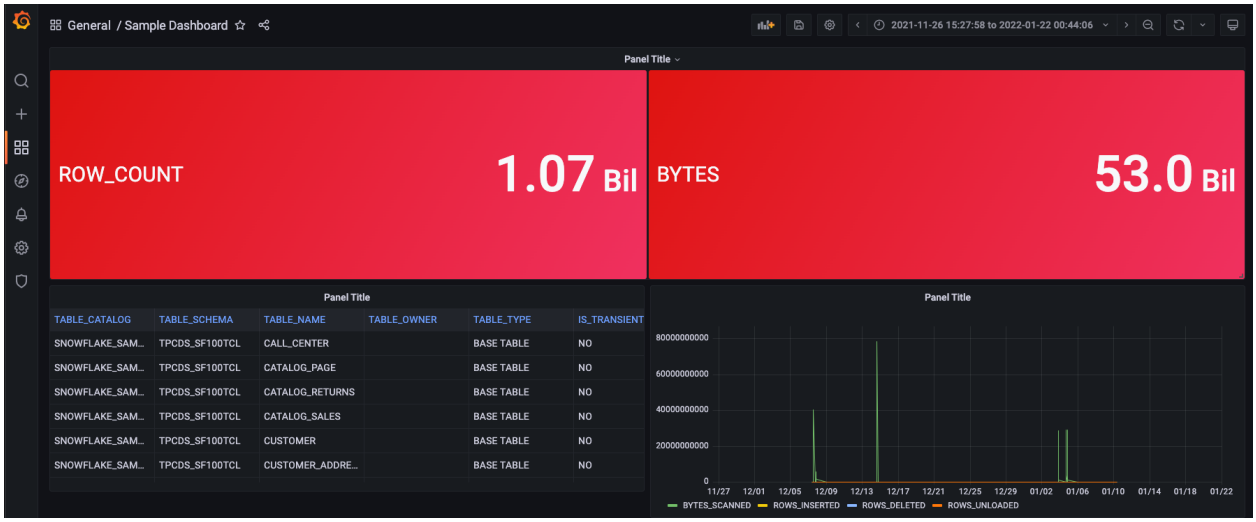

Figure 6: Grafana integration with Snowflake

Datadog provides monitoring for databases, applications, and other services. It can connect to Snowflake through the Datadog agent and is able to extract metrics such as credit usage, billing, and query metrics. More details about integrating Snowflake can be found in Datadog's documentation.

Splunk analyzes log and event data that is generated by various applications and systems. Splunk's DBConnect utility allows DevOps teams to easily establish a connection between Snowflake and Splunk via JDBC. With this, Snowflake logs and events can be piped into Splunk to allow proactive monitoring.

New Relic provides a Snowflake integration which gives DevOps teams an overview of areas such as performance, cost, and security in Snowflake. New Relic alerts allow you to set customizable alert policies so that DevOps teams get quickly informed about events such as when queries are queuing at a Snowflake warehouse.

Partners that have used Snowflake to build their observability platform

Observe is a SaaS observability platform focused on providing discrete technologies for log analytics, metrics monitoring, and application monitoring. The company built its platform using Snowflake because of its ability to handle relational data as well as semi-structured and time-series data. This meant Observe could process all event type data in a single place and use this to quickly build up its platform.

Panther Labs is a cybersecurity company specializing in detection and response. They are building an SIEM solution on top of Snowflake and are focusing on aggregating

security-related data and logs. This enables continuous monitoring in which logs are analyzed in real time to identify suspicious activity that could indicate a breach. Learn more about Panther and check out its cloud-native SIEM listing in Snowflake Data Marketplace.

Conclusion

Snowflake offers many capabilities that DevOps teams can use to empower observability. When considering your own data observability platform, look for the following:

- Ability to extract a wealth of key observability metrics such as query history and performance
- Native functions that enable proactive actions, such as external functions and Snowpipe error notification
- Enhanced troubleshooting with the inclusion of easy-to-use visualization capabilities through Snowsight
- Easy partner integration with an ecosystem of third-party observability platforms