Confluent Cloud for Apache Flink (CCAF) Tableflow AWS Glue Snowflake Kickstarter

Data practitioners are entering a golden era-a time defined by groundbreaking possibilities and transformative innovation. In the early days, building data warehouses required enormous intellectual and financial investments. We carefully engineered and maintained limited conforming dimensions and facts, continuously adapting to meet evolving business needs. Transferring data from source to target not only incurred high costs but also stripped away vital context, which had to be painstakingly rebuilt to derive actionable insights.

As we evolved to data lakes, many challenges persisted: maintenance overhead, slow adaptability to surging data demands, and the constant struggle to preserve context. With the burgeoning tide of ML and AI, the stakes have escalated even further. Yet, these challenges are paving the way for unprecedented opportunities for innovation and efficiency. Today, every obstacle is a stepping stone toward a more agile, insightful, and future-ready data landscape.

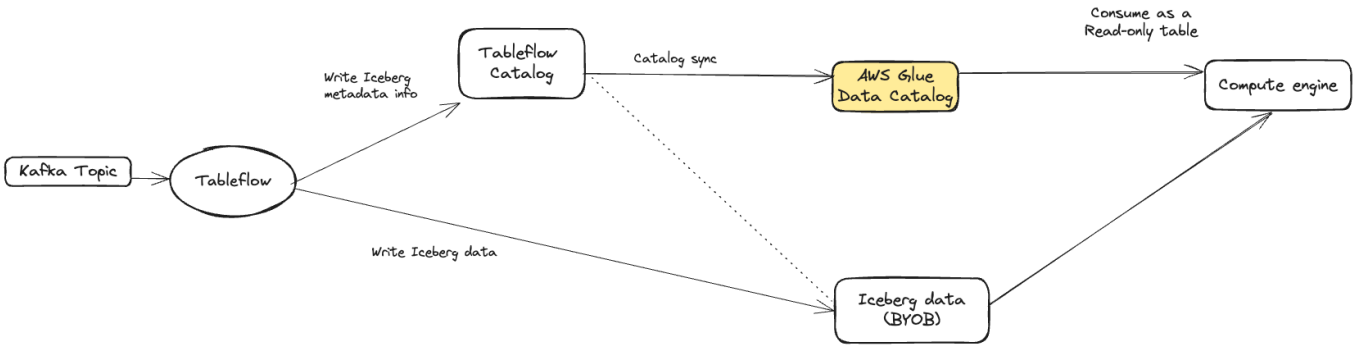
On March 19, 2025, Confluent proudly announced the general availability of Tableflow for Apache Iceberg, marking a transformative milestone for data warehousing and data lakes. This monumental release redefines data management by seamlessly addressing the complexities of modern data infrastructures. Leveraging the unparalleled power of our fully managed open-source trifecta—Apache Kafka, Apache Flink, and Apache Iceberg—we now deliver a unified solution that adeptly serves both operational and analytical data needs.

Welcome to the forefront of the data revolution, where every challenge is an opportunity and innovation knows no bounds.

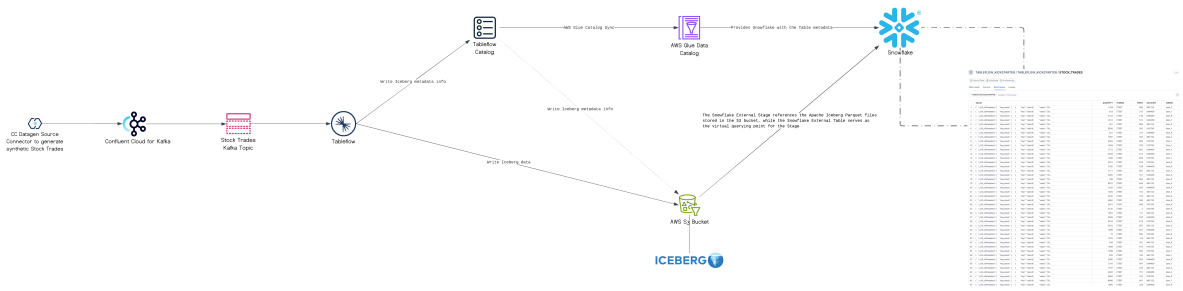
1.0 The Impetus

The driving force behind this project is the need to **simplify** and **automate** the process of setting up a Confluent Cloud environment with Tableflow for Apache Iceberg, AWS S3 Bucket, AWS Glue Data Catalog, and Snowflake Database. The goal is to eliminate the *manual steps* involved in configuring these components, allowing data practitioners like you to focus on building data products instead of managing infrastructure.

Turned this picture:



Into reality:



With the assistance Terraform, a powerful tool for infrastructure as code (IaC), you can define and manage your infrastructure using declarative configuration files. This project utilizes Terraform to automate the setup of Confluent Cloud, AWS S3 Bucket, AWS Glue Data Catalog, and Snowflake, ensuring a ***consistent*** and ***repeatable*** deployment process.

2.0 Let's Get Started!

These are the steps

1. Take care of the local environment prequisities listed below:
    - You need to have the following installed on your local machine:
        - AWS CLI version 2
        - Confluent CLI version 4 or higher
        - Terraform CLI version 1.12.2 or higher
2. Get your Confluent Cloud API key pair, by execute the following Confluent CLI command to generate the Cloud API Key:

    > Click here to learn why you need it.

    ```
    confluent api-key create --resource "cloud"
    ```

The API Key pair allows Terraform to provision, manage, and update Confluent Cloud resources as defined in your infrastructure code, maintaining a secure, automated deployment pipeline.

3. Clone the repo:

```
git clone https://github.com/j3-signalroom/ccaf-tableflow-aws_glue-snowflake-kickstarter.git
```

4. Apart of the Terraform configurations, is the `snowflake_user_rsa_key_pairs_rotation`, the `iac-snowflake-user-rsa_key_pairs_rotation-tf_module` Terraform module to automate the creation and rotation of RSA key pairs for a Snowflake service account user. It leverages a specialized AWS Lambda function, known as the `iac-snowflake-user-rsa_key_pairs_generator-lambda`, to automate the generation and rotation of RSA key pairs. The module allows users to define rotation intervals (e.g., every 30 days since the last key generation) to enhance security by regularly renewing cryptographic credentials. Additionally, it integrates seamlessly with AWS Secrets Manager to securely store and manage the generated key pairs, ensuring that the keys remain protected and easily accessible for Snowflake authentication without manual intervention.

2.1 DevOps in Action: Running Terraform Locally

Install the Terraform CLI on your local machine, and make sure you have an HCP Terraform account to run the Terraform configuration. Learn how to set up Terraform Cloud for local use by clicking here.

Then run the following command to set up the Terraform configuration locally. This command will create a Confluent Cloud environment with a Kafka Cluster configured for Tableflow, AWS Secrets Manager, an AWS S3 bucket, AWS Glue Data Catalog, and Snowflake Database:

> **Note**: *The script and this project in general assumes your hyperscaler (i.e., cloud provider) is **AWS**. Moreover, that it is expected the AWS account is configured with SSO (Single Sign On) support.*

```
deploy.sh <create | delete> --profile=<SSO_PROFILE_NAME> \
                            --confluent-api-key=<CONFLUENT_API_KEY> \
                            --confluent-api-secret=<CONFLUENT_API_SECRET> \
                            --snowflake-warehouse=<SNOWFLAKE_WAREHOUSE> \
                            --day-count=<DAY_COUNT> \
                            --number-of-api-keys-to-retain=<NUMBER_OF_API_KEYS_TO_RETAIN>
```

Argument placeholder	Replace with
`<SSO_PROFILE_NAME>`	your AWS SSO profile name for your AWS infrastructue that host your AWS Secrets Manager.
`<CONFLUENT_API_KEY>`	your organization's Confluent Cloud API Key (also referred as Cloud API ID).

Argument placeholder	Replace with
`<CONFLUENT_API_SECRET>`	your organization's Confluent Cloud API Secret.
`<SNOWFLAKE_WAREHOUSE>`	the Snowflake warehouse (or "virtual warehouse") you choose to run the resources in Snowflake.
`<DAY_COUNT>`	how many day(s) should the API Key be rotated for.
`<NUMBER_OF_API_KEYS_TO_RETAIN>`	specifies the number of API keys to create and retain.

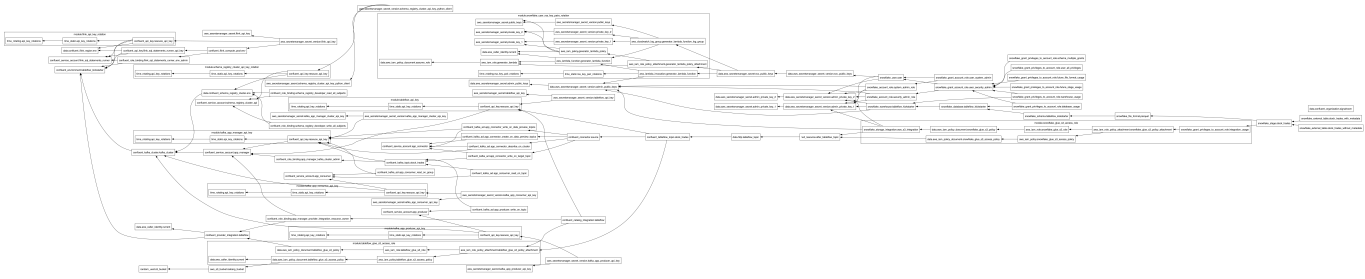
To learn more about this script, click here.

After a successfuly run of the script, here's what you can expect:

- A Confluent Cloud environment featuring a Kafka Cluster, fully equipped with pre-configured example Kafka topics—ready to power your data streaming needs.
- AWS Secrets Manager securely stores API Key Secrets for the Kafka Cluster.
- Configure the Datagen Source Connector Kafka Topics for Tableflow.
- An AWS S3 bucket with a dedicated folder, named after the Kafka Cluster ID, that serves as the landing zone for Apache Iceberg Tables populated by the Datagen Source Connector.
- An AWS Glue Data Catalog ensures seamless integration with the S3 bucket and enables efficient data discovery.
- A Snowflake Database, where the data from the S3 bucket will be ingested and transformed into a Snowflake Table.

2.2 Visualizing the Terraform Configuration

Below is the Terraform visualization of the Terraform configuration. It shows the resources and their dependencies, making the infrastructure setup easier to understand.



> **To fully view the image, open it in another tab on your browser to zoom in.**

When you update the Terraform Configuration, to update the Terraform visualization, use the `terraform graph` command with Graphviz to generate a visual representation of the resources and their dependencies. To do this, run the following command:

```
terraform graph | dot -Tpng > .blog/images/terraform-visualization.png
```

3.0 Resources

- Shift Left: Unifying Operations and Analytics With Data Products eBook

3.1 Confluent Cloud for Apache Kafka (CCAK)

- Datagen Source Connector for Confluent Cloud

3.2 Confluent Cloud for Apache Flink (CCAF)

- Stream Processing with Confluent Cloud for Apache Flink

3.3 Tableflow for Apache Iceberg

- Tableflow in Confluent Cloud
- Terraforming Snowflake
- Terraform Provider Confluent Tableflow Examples Configuration
- Learn more about Apache Iceberg

3.4 AWS Glue Data Catalog

- Data discovery and cataloging in AWS Glue

3.5 Snowflake

- Snowflake Create Storage Integration
- Snowflake Terraform Registry
- Option 1: Configuring a Snowflake storage integration to access Amazon S3

4.0 Important Note(s)

- Known Issue(s)