*GCP Dataform (Pipelines) – Bayer*

# Introduction:

***Dataform (now Pipelines)*** is a free serverless service for data analysts to develop and deploy tables, incremental tables & views, test, version control, and schedule complex SQL workflows for data transformation in BigQuery. Dataform offers a web environment for SQL workflow development, connection with GitHub, GitLab, Azure DevOps Services, and Bitbucket, continuous integration, continuous deployment, and workflow execution.

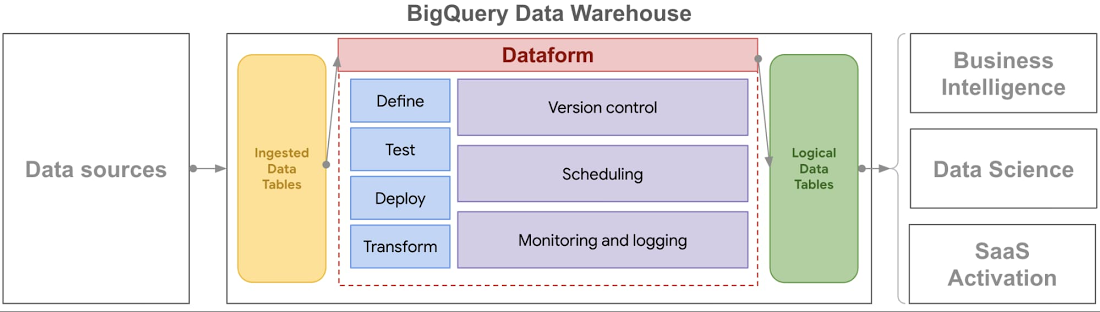
***Dataform*** lets you manage data transformation in the Extraction, Loading, and Transformation (ELT) process for data integration. After raw data is extracted from source systems and loaded into BigQuery, Dataform helps you to transform it into a well-defined, tested, and documented suite of data tables.

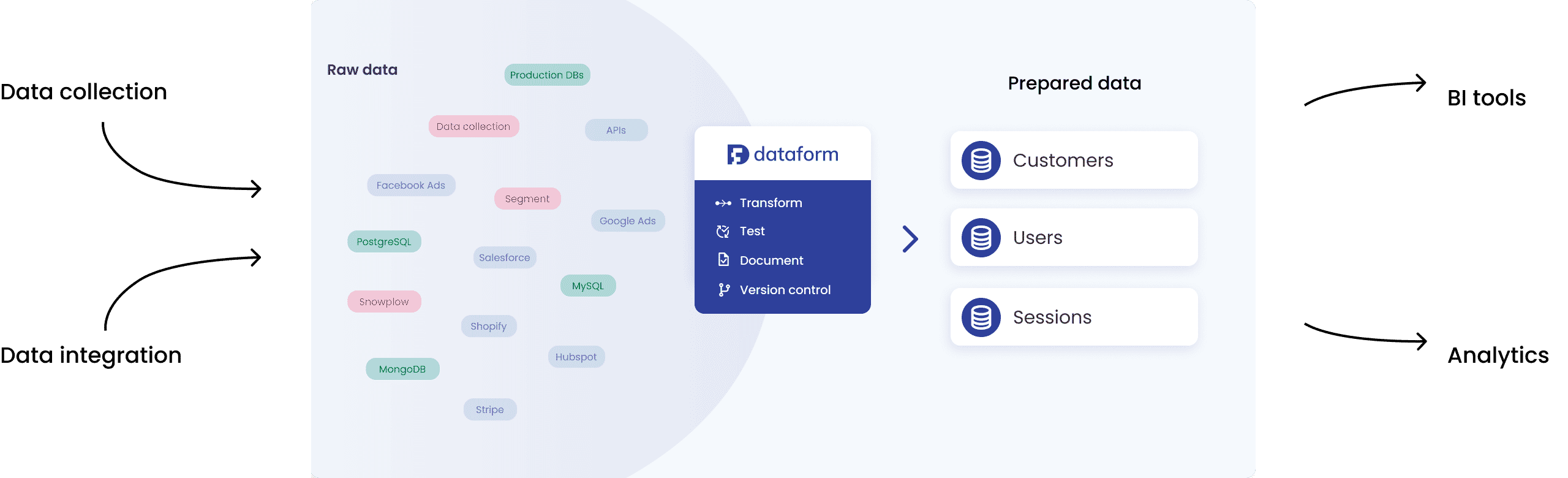
***Dataform*** lets you perform the following data transformation actions:

* Develop and execute SQL workflows for data transformation.
* Collaborate on SQL workflow development through Git.
* Declare source data and manage table dependencies.
* View a visualization of the dependency tree of your SQL workflow.
* Manage data with SQL code in a central repository.
* Reuse code with JavaScript.
* Test data correctness with quality tests on source and output tables.
* Version control SQL code.
* Document data tables inside SQL code.

***Dataform*** provides an open source data modeling framework, consisting of Dataform core and Dataform CLI, that you can use outside Google Cloud.

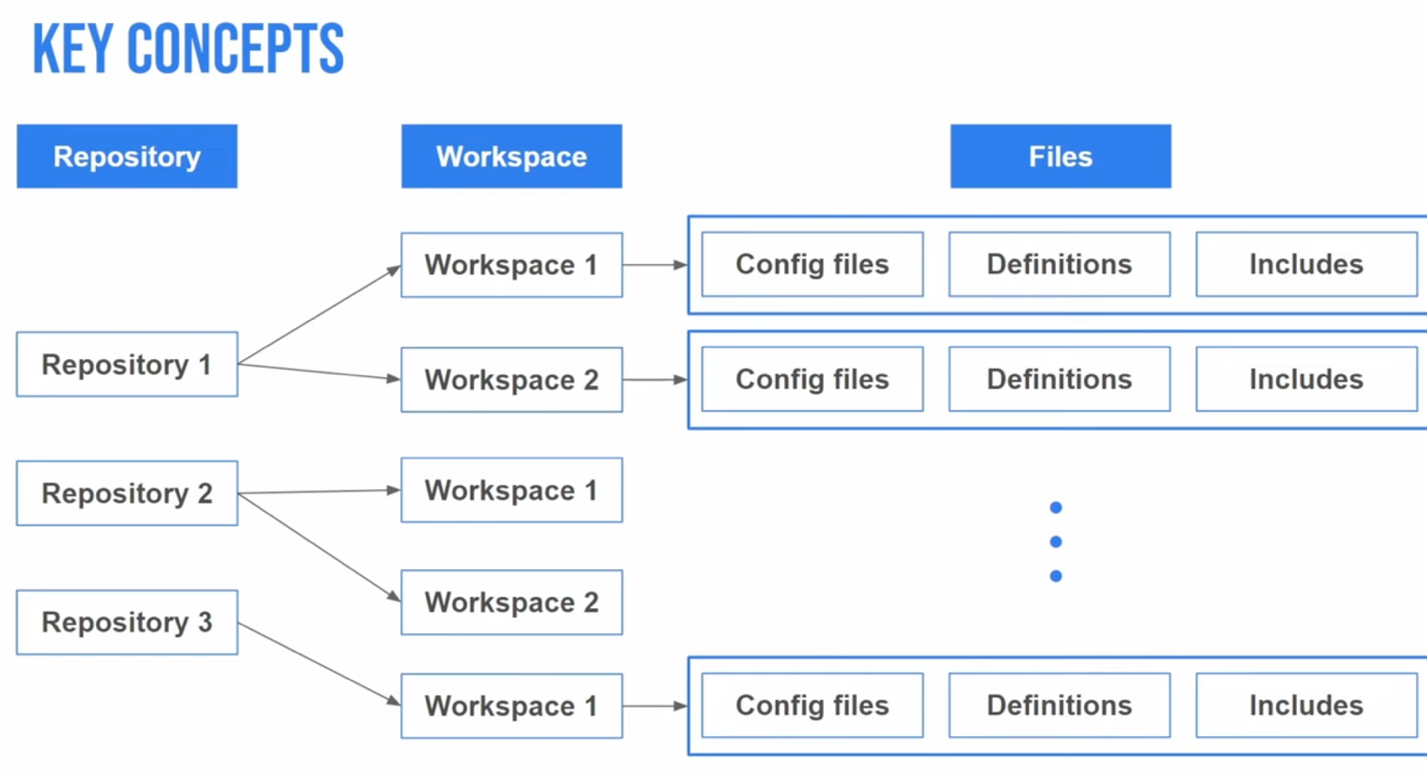
*Dataform Core* is an open source meta-language to create SQL tables and workflows in BigQuery. Dataform Core extends SQL by providing a dependency management system, automated data quality testing, and data documentation. Using Dataform Core, data teams can build scalable SQL data transformation pipelines following software engineering best practices, like version control and testing.





This document is divided into 2 parts. The first part deals with the initializing & setup of Dataform at Bayer i.e. implementation and the second part deals with the different features of Dataform workflows (pipelines) and their usage i.e. deployment.

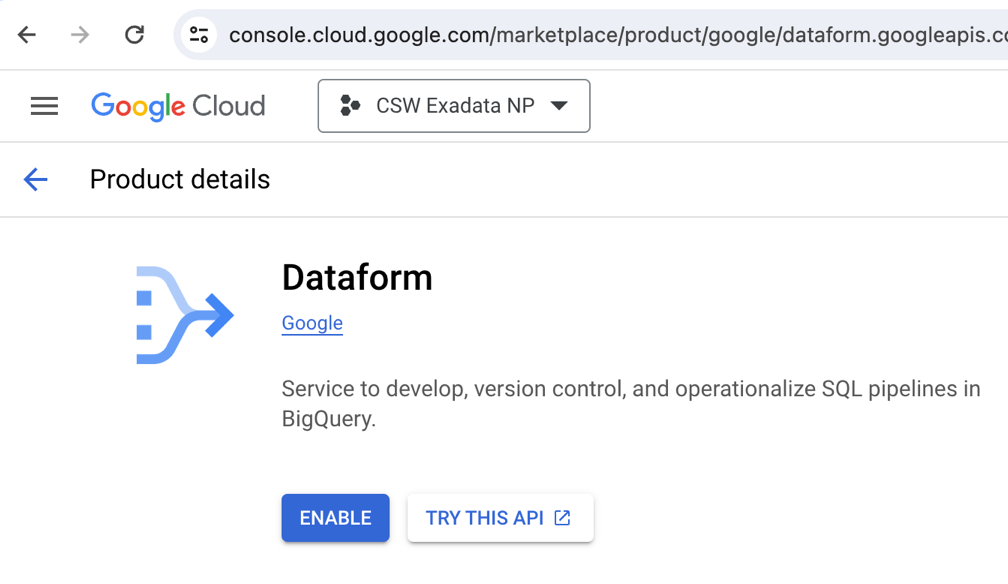
To implement & deploy Dataform for your team, you first need to understand the key concepts of BigQuery Dataform in detail, as shown below:



### Note: [Documentation home page](https://cloud.google.com/dataform)

# Implementing dataform at bayer:

To implement Dataform for your team, you first need to enable the Dataform API in the GCP project where your core BigQuery (BQ) tables are located as shown below.



* 1. service account:

At Bayer however, in accordance with policies & standards, instead of using the default Dataform service account that comes up once the API is enabled, you must add it to the IAM principal for that project and use that principal as the Dataform SA for all intents & purposes.

Also note that, Dataform core & SA (service account) used in a specific **NP** (non-prod) project should ONLY have access to BQ tables or source data from other NP projects. Similarly, Dataform core & SA for a specific **Prod** project should only be granted access to BQ tables or source data from other Prod environments.

Dataform must be provided with certain access privileges to function without error. The list of roles & permissions that need to be granted to the Dataform SA are listed as shown below:

A screenshot of a computer

AI-generated content may be incorrect.

Dataform can also be used to control access to individual tables as well as with IAM. You can configure VPC controls & use customer-managed encryption keys in Dataform as well. Please see link in NOTE.

### Note: [click here](https://cloud.google.com/dataform/docs/repositories)

* 1. repository:

Each Dataform project is stored in a repository. A Dataform repository houses a collection of JSON configuration files, SQLX files, and JavaScript files. Dataform repositories contain the following types of files:

* **Config files**

Config JSON or SQLX files let you configure your SQL workflows. They contain general configuration, execution schedules, or schema for creating new tables and views.

* **Definitions**

Definitions are SQLX and JavaScript files that define new tables, views, and additional SQL operations to run in BigQuery.

* **Includes**

Includes are JavaScript files where you can define variables and functions to use in your project.

When you create your first Dataform repository, Dataform automatically generates a default service account, which does not have any BigQuery roles or permissions, to interact with BigQuery on your behalf.

A screenshot of a computer

AI-generated content may be incorrect.

A Bayer team (functional org) can have any number of repositories depending on how they would like to organize their sub-teams. Each repository could correspond to a team or their respective project as shown above.

Best practice dictate that repositories should be divided accordingly:

* 1 repository per a development team.
* 1 repository per domain, for example, sales, marketing, or logistics.
* 1 central repository and 1 repository per domain that uses the contents of the central repository as data sources.

The corresponding Git repo will be configured & connected to the repositories that have been created and will be located within the organizations’ central Git repo. Repositories will show up as branches.

A screenshot of a computer

AI-generated content may be incorrect.

### Note: [click here](https://cloud.google.com/dataform/docs/repositories)

* 1. Connecting DF repo to Git repository:

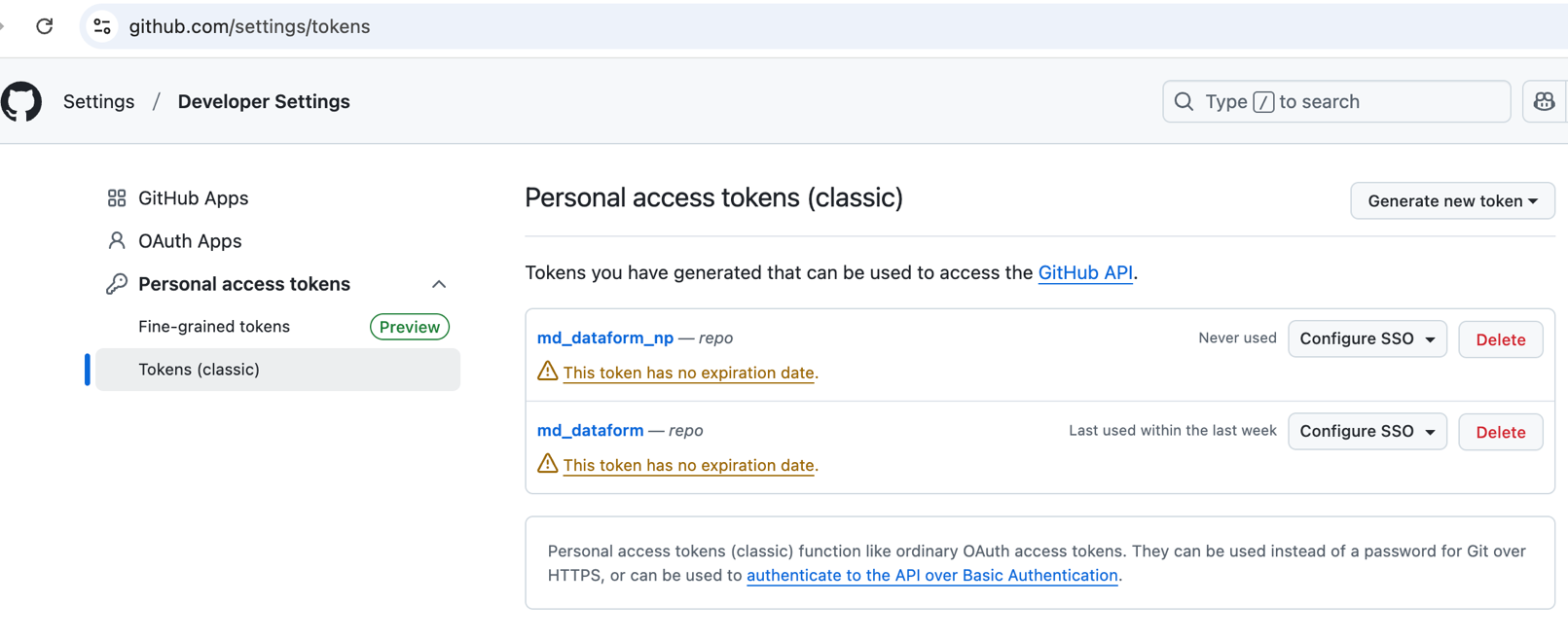
There are 2 ways to connect your Bayer GCP Dataform repository with a third-party Git repository:

1. Through SSH
2. Through HTTPS (Recommended)

To connect your DF repo with a Git Repo through SSH, [click here](https://cloud.google.com/dataform/docs/connect-repository).

To connect your DF repo with a Git Repo through HTTPS, follow these steps:

1. In GitHub, create a [fine-grained personal access token](https://github.blog/2022-10-18-introducing-fine-grained-personal-access-tokens-for-github/) or a [classic personal access token](https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/creating-a-personal-access-token#about-personal-access-tokens). If your organization uses SAML single sign-on (SSO), [authorize the token](https://docs.github.com/en/enterprise-cloud@latest/authentication/authenticating-with-saml-single-sign-on/authorizing-a-personal-access-token-for-use-with-saml-single-sign-on).

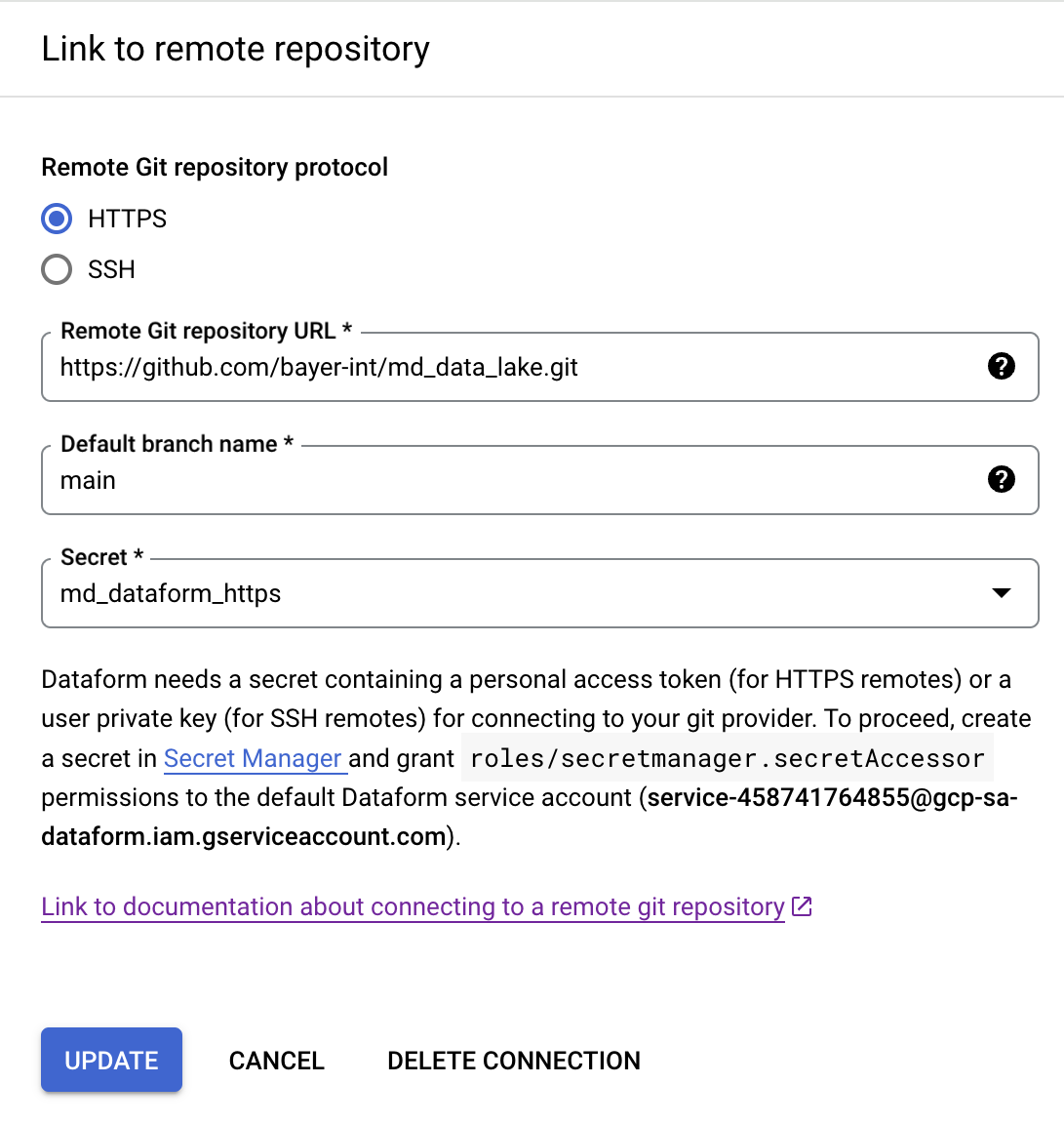


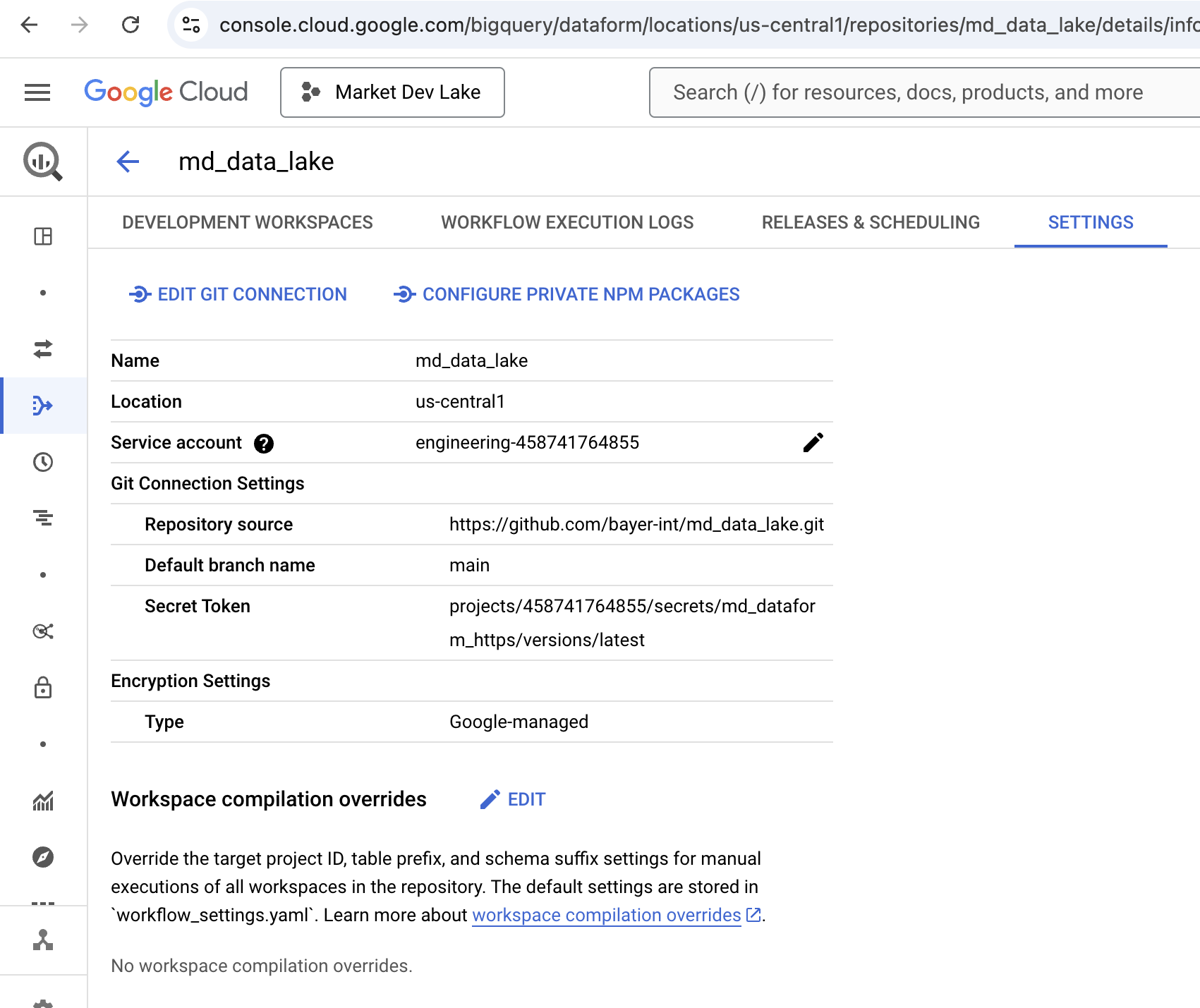
1. In Secret Manager, [create a secret](https://cloud.google.com/secret-manager/docs/creating-and-accessing-secrets#create) containing the personal access token of your remote repository and [Grant access to the secret to your default Dataform service account](https://cloud.google.com/secret-manager/docs/manage-access-to-secrets).

A screenshot of a computer

AI-generated content may be incorrect.

1. In the Google console, go to Dataform, select the DF repo that you want to connect to the remote repo. On the repository page, click **Settings > Connect with Git**. In the **Link to remote repository** pane, in the **Remote Git repository URL** field, enter the URL of the remote Git repository, ending with **.git**. The URL of the remote Git repository cannot contain usernames or passwords. In the **Default remote branch name** field, enter the name of the main development branch of the remote Git repository. In the **Secret** drop-down, select your secret for the remote Git repository. Click **Link.**
2. You could also edit your Git connection.





### Note: [click here](https://cloud.google.com/dataform/docs/repositories)

# DEPLOYING dataform at bayer:

To deploy Dataform for your team, you first need to understand the key concepts of BigQuery Dataform in detail, as shown below:

* 1. development workspace & REPO STRUCTURE:

In Dataform, you make changes to files and directories inside a development workspace. A development workspace is a virtual, editable copy of the contents of a Git repository. Dataform preserves the state of files in your development workspace between sessions. Each workspace in a repository could correspond to a specific team project.

An initialized development workspace contains the following directories and files:

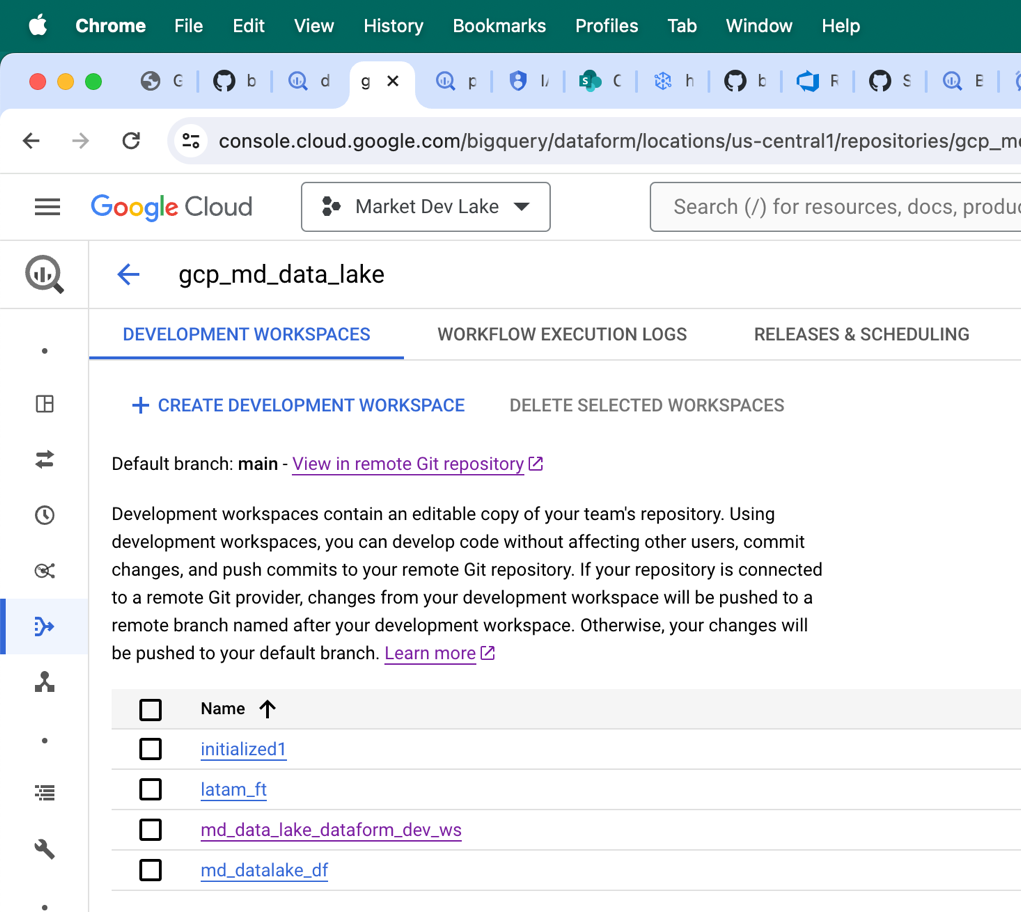
* **definitions/**: a directory for asset definitions, in Dataform core or JavaScript. All subdirectories must be listed under this directory.
* **includes/**: an empty directory for scripts and variables that you can reuse across the repository.
* **dataform.json**: the default Dataform configuration file containing the Google Cloud project ID and BigQuery schema to publish assets in. You can override the default settings to customize them to your needs, but it's not a requirement to begin using Dataform.
* **package.json**: the default Dataform dependencies configuration file with the latest version of @dataform/core. You can use this file to import packages.
* **definitions/sample.sqlx**: a sample SQLX file to help you get started.

Recommended structure for structuring repository files (.sqlx & .json files) in the definitions directory to reflect the stages of your workflow, is as follows:

* *sources*, storing data source declarations
* *intermediate/staging*, storing data transformation logic
* *output/reporting*, storing definitions of output tables
* Optional: extras, storing additional files

A screenshot of a computer

AI-generated content may be incorrect.



### Note: [click here](https://cloud.google.com/dataform/docs/workspaces)

* 1. SQL workflow:

You can develop SQL workflow actions by using Dataform core with SQLX and JavaScript, or exclusively with JavaScript. Each element of a Dataform SQL workflow, such as a table or assertion, corresponds to an action that Dataform performs in BigQuery. For example, a table definition file is an action of creating or updating the table in BigQuery.

In a Dataform workspace, you can develop the following SQL workflow actions:

* Data Source declarations: Declarations of BigQuery data sources that let you reference these data sources in Dataform table definitions and SQL operations.
* Tables and views: Tables that you create in Dataform based on the declared data sources or other tables in your SQL workflow. Dataform supports the following table types: table, incremental table, view, and materialized view.
* Custom SQL operations: SQL statements that Dataform runs in BigQuery as they are, without modification.
* Includes: JavaScript files with definitions of variables and functions that you can reuse across your SQL workflow.
* Data quality tests, called ***assertions***: Data quality test queries that you can use to validate table data. Dataform runs assertions every time it updates your SQL workflow and it alerts you if any assertions fail.
* Incremental tables
* Table partitions and clusters
* Dependencies between actions
* Documentation of tables
* BigQuery labels & policy tags
* Dataform table tags
  1. Data source declarations:

You can declare any BigQuery table type as a data source in Dataform. Declaring BigQuery data sources that are external to Dataform lets you treat those data sources as first-class Dataform objects. After you declare a data source, you can reference or resolve it in the same way as any other table in Dataform.

### Note: [click here](https://cloud.google.com/dataform/docs/declare-source)

**Example:**

config {

type: "declaration",

database: "bcs-market-dev-lake",

schema: "global\_velocity",

name: "allocate\_environment\_field\_reservation",

description: "source table view from csw core."

}

* 1. Dependencies:

You can define a dependency relationship between objects of a SQL workflow. In a dependency relationship, the execution of the dependent object depends on the execution of the dependency object. This means that Dataform executes the dependent after the dependency. You define the relationship by declaring dependencies inside the SQLX definition file of the dependent object. The dependency declarations make up a dependency tree of your SQL workflow that determines the order in which Dataform executes your SQL workflow objects.

There are 2 ways to declare dependencies:

* Declare a dependency by using the Dataform core ref function to reference the dependency in a SELECT statement.
* Declare a list of dependencies in the config block of a SQLX definition file.

### Note: [click here](https://cloud.google.com/dataform/docs/dependencies)

* 1. Custom sql operations:

Dataform can execute custom SQL operations that don't fit into the Dataform model of publishing a table or writing an assertion. You can define custom SQL commands for Dataform to execute in BigQuery. You can define a custom SQL operation in a SQLX file of **type: operations**. You can write any BigQuery SQL statement in an operations file. Dataform runs your custom SQL operations in BigQuery without modification.

### Note: [click here](https://cloud.google.com/dataform/docs/custom-sql)

* 1. execution tags:

To organize components of your SQL workflow into collections, you can add custom tags to SQLX files of the following types:

1. table
2. view
3. incremental
4. assertion
5. operations
6. Tables:

In Dataform, a table is one of the types of objects that make up a SQL workflow. You can create tables that reference data from the data sources declared for your SQL workflow, or other tables in your SQL workflow. Dataform compiles your table definitions into SQL in real time. When you trigger execution, Dataform executes the SQL code and creates your defined tables in BigQuery.

* You can create the following table types in a type: "table" SQLX file:
* **table**: a regular table.
* **incremental**: an incremental table.
* **view**: a table view.
* **materialized**: a materialized table view.
* You can also define table partitions and clusters.
* To keep a record of the purpose of a table or its relation to other tables in your SQL workflow, you can add documentation to the table or its selected columns.
* To test data in a table against specific conditions, you can create data quality test queries called assertions. Dataform runs assertions every time it updates your SQL workflow and alerts you if any assertions fail.
* To override default table settings, such as database or schema, and disable table creation, or execute a SQL statement before or after table creation, you can configure additional table settings.
* To organize your tables in BigQuery after you execute them, you can add BigQuery labels.
* To restrict data access at the table column level, you can add BigQuery policy tags.
* In addition to defining tables in a type: "table" SQLX file, you can create empty tables by defining a custom SQL query in a type: "operations" SQLX file. You might want to create an empty table so that a different service can populate it with data.

### Note: [click here](https://cloud.google.com/dataform/docs/tables)

1. INCREMENTAL TABLES:

When you define an incremental table, Dataform builds the incremental table from scratch only for the first time. During subsequent executions, Dataform only inserts or merges new rows into the incremental table according to the conditions that you configure.

Dataform inserts new rows only into columns that already exist in the incremental table. If you make changes to the incremental table definition query — for example, add a new column — you must rebuild the table from scratch. To do so, the next time you trigger an execution of the table, select the **Run with full refresh** option.

USE CASES FOR INCREMENTAL TABLES:

* *Performance optimization* -- only process new records instead of reprocessing the entire table.
* *Latency reduction* -- reduce the downstream latency of the output tables.
* *Daily Snapshots* -- create daily snapshots of table data.
* Process a subset of rows in an incremental table.
* Merge rows in an incremental table.
* Filter rows in an incremental table.
* Avoid full table scans when ingesting from a partitioned table.
* Configure Dataform for Slowly Changing Dimensions:

The  [Slowly changing dimensions package](https://github.com/dataform-co/dataform-scd) contains common data models for creating type 2 slowly changing dimensions tables from mutable data sources in Dataform.

Slowly changing dimensions tables are incremental tables that contain data that can change unpredictably, not on a regular schedule, such as customers or products. In a type 2 slowly changing dimensions table, new data is appended in a new row without overwriting existing table rows. Table history is preserved in multiple records for a given key in the slowly changing dimension key. Each record has a unique key.

The Slowly changing dimensions package creates the following relations in BigQuery for a given NAME:

**NAME** - a view with **scd\_valid\_from** and **scd\_valid\_to** fields.

**NAME\_updates** - an incremental table that stores the change history of the source table.

Dataform updates slowly changing dimensions every time it executes a slowly changing dimensions incremental table. You might want to schedule slowly changing dimensions table to run every day or every hour, depending on the granularity of changes you want to capture.

### Note: [click here](https://cloud.google.com/dataform/docs/incremental-tables)

1. aSSERTions:

An assertion is a data quality test query that finds rows that violate one or more rules specified in the query. If the query returns any rows, the assertion fails. Dataform runs assertions every time it updates your SQL workflow and it alerts you if any assertions fail. Dataform automatically creates views in BigQuery that contain the results of compiled assertion queries. You can add assertions as dependencies.

You can create assertions in the following ways:

* Add built-in assertions to the config block of a table.

You can add built-in assertions to the config block of a table and specify their conditions.

* Add manual assertions in a separate SQLX file.

You manually write custom assertions in a separate SQLX file for advanced use cases or for datasets not created by Dataform.

You can use the 3 built-in assertions provided by Dataform:

* nonnull
* rowConditions
* uniqueKeys

### Note: [click here](https://cloud.google.com/dataform/docs/assertions)

* 1. Schedule executions with workflow configurations:

To schedule Dataform executions of all or selected SQL workflow actions in BigQuery, you can create workflow configurations. In a workflow configuration, you select a compilation release configuration, select SQL workflow actions for execution, and set the execution schedule.

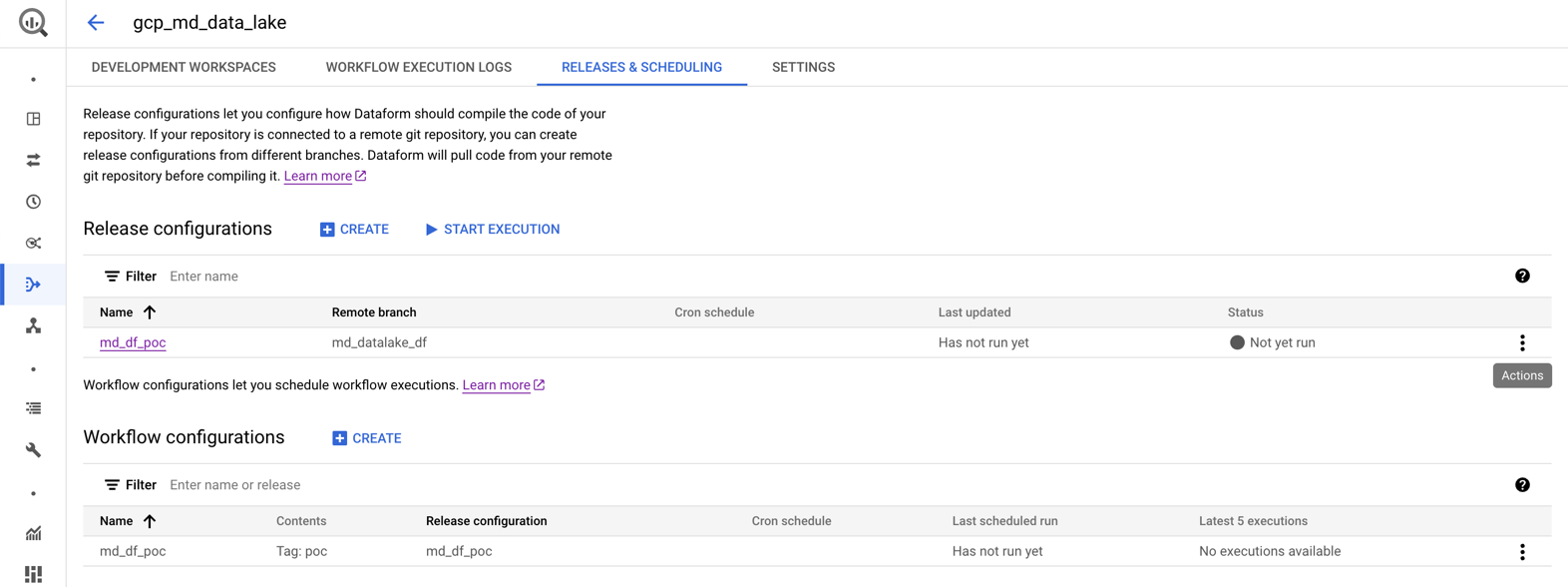
Then, during a scheduled execution of your workflow configuration, Dataform deploys your selection of actions from the latest compilation result in your release configuration to BigQuery. You can also manually trigger execution of a workflow configuration with the Dataform API workflowConfigs.

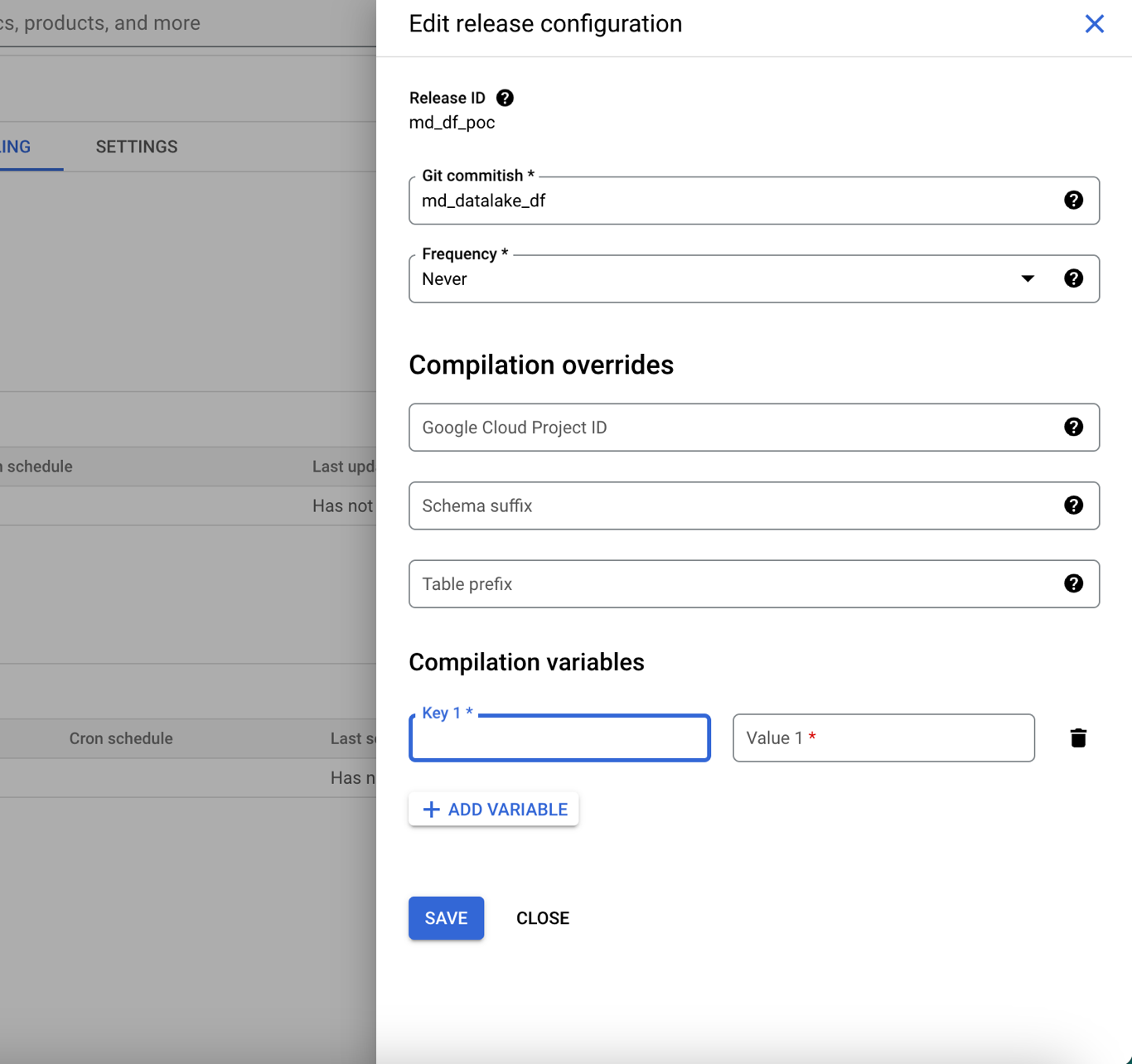
A Dataform workflow configuration contains the following execution settings:

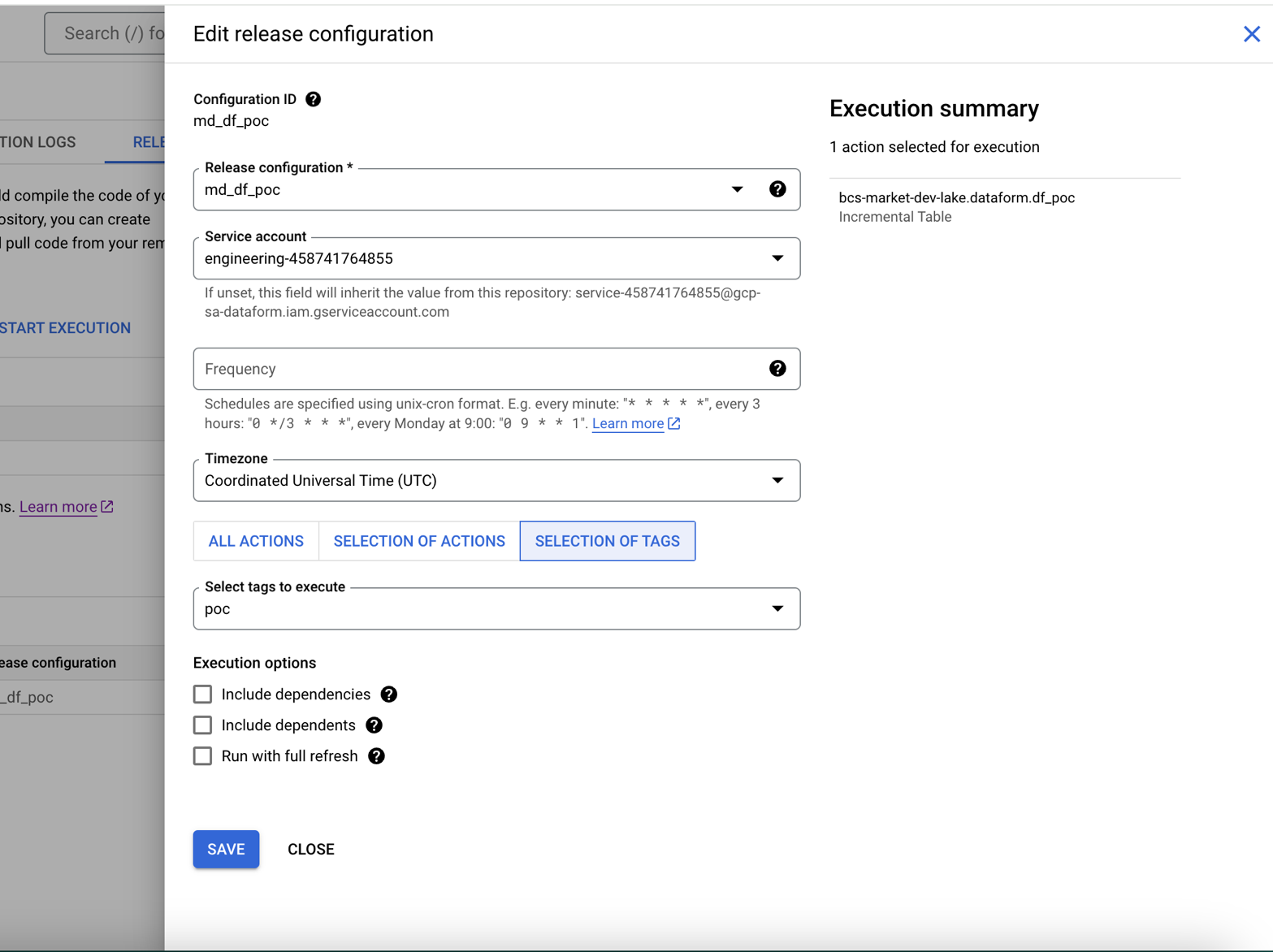
* ID of the workflow configuration
* Release configuration
* Service account

Service account associated with the workflow configuration. You can select the default Dataform service account, a service account associated with your Google Cloud project, or manually enter a different service account. By default, workflow configurations use the same service accounts as their repositories.

* SQL workflow actions to be executed:
  + All actions
  + Selection of actions
  + Selection of tags
* Execution schedule and time zone







* 1. useful links & sample projects:
* Enable [scripting](https://cloud.google.com/dataform/docs/develop-workflows-js) and code re-use with a JavaScript API.
* Import [pre-defined packages](https://dataform-co.github.io/dataform/docs/packages), or create your own.
* View the [Dataform Core reference](https://cloud.google.com/dataform/docs/reference/dataform-core-reference) & [Dataform configs reference](https://dataform-co.github.io/dataform/docs/configs-reference).
* <https://github.com/GoogleCloudPlatform/marketing-data-engine-dataform>.
* <https://github.com/wintermi/movielens-dataform>.
* <https://github.com/wintermi/fashion-dataform>.
* <https://github.com/G2H/dataform-stackoverflow>.
* <https://github.com/karcot1/dataform_deployment_sample>.

*Note: this readme can also be viewed on*[*https://dataform-co.github.io/dataform*](https://dataform-co.github.io/dataform)*.*