Kedro Snowflake Plugin

Release 0.2.1

GetInData

Jun 20, 2023

CONTENTS:

CHAPTER

ONE

INTRODUCTION

1.1 What are Snowflake / Snowflake Tasks / Snowpark SDK?

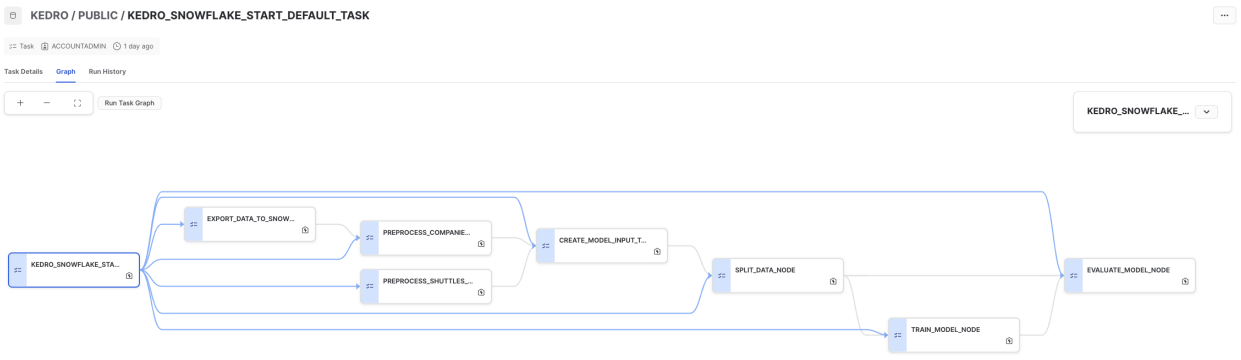
Snowflake is a cloud-based data warehousing platform that provides scalable storage and analytics capabilities. It is designed to be highly flexible and allows users to store and process large amounts of data across multiple regions and clouds. Snowflake's unique architecture separates compute resources from storage, enabling users to scale up and down compute resources as needed without affecting their data.

Snowflake Tasks are a feature within Snowflake that allows users to define and manage complex data processing workflows. A task consists of one or more SQL statements and can be triggered automatically based on a predefined schedule or manually by a user. Tasks can be used to automate data integration and transformation processes, making it easier to maintain and manage complex data pipelines.

The Snowpark Python SDK is a library that allows users to write Python code that interacts with Snowflake. With the Snowpark Python SDK, users can create and manage objects such as tables, views, and stored procedures, as well as execute SQL queries directly from Python. This enables users to leverage the full power of Python to create complex data processing workflows that seamlessly integrate with Snowflake.

1.2 Why to integrate Kedro project with Snowflake?

The Kedro Snowflake plugin provides an efficient and scalable way to execute Kedro pipelines directly within Snowflake, leveraging Snowpark Python SDK. By using the plugin, users can seamlessly translate Kedro pipelines into Snowflake tasks and run them without the need for external systems. This makes it an ideal solution for users who need to scale up Kedro Machine Learning pipelines in Snowflake, with direct access to data.



Kedro pipeline in Snowflake

CHAPTER

TWO

INSTALLATION GUIDE

2.1 Prerequisites

- Python 3.8 is a must - this is enforced by the `snowflake-snowpark-python` package. Refer to Snowflake documentation for more details.
- A tool to manage Python virtual environments (e.g. venv, conda, virtualenv). Anaconda is recommended by Snowflake.
- Kedro is fixed for now at version `<0.18.9` due to data set errors that appear in later versions.

2.2 Plugin installation

2.2.1 Install from PyPI

Install the plugin (it automatically installs Kedro in a supported version)

```
$ pip install "kedro-snowflake>=0.1.0"
```

2.2.2 Install from sources

You may want to install the develop branch which has unreleased features:

```
pip install git+https://github.com/getindata/kedro-snowflake.git@develop
```

2.3 Available commands

You can check available commands by going into project directory and running:

```
kedro snowflake
kedro snowflake [OPTIONS] COMMAND [ARGS]...

Options:
  -e, --env TEXT  Environment to use.
  -h, --help      Show this message and exit.
```

(continues on next page)

(continued from previous page)

```
Commands:
  init  Creates basic configuration for Kedro Snowflake plugin
  run   Runs the pipeline using Snowflake Tasks
```

CHAPTER

THREE

QUICKSTART

Before you start, make sure that you have access to Snowflake account and prepare the following information:

- Snowflake Username
- Snowflake Password
- Snowflake Account Name
- Snowflake Warehouse Name
- Snowflake Database Name
- Snowflake Schema Name
- Snowflake password (you will store it locally in an environment variable)

You will also need:

- Python 3.8 (must-have - this is enforced by the *snowflake-snowpark-python* package. Refer to Snowflake documentation for more details.
- A tool to manage Python virtual environments (e.g. venv, conda, virtualenv). Anaconda is recommended by Snowflake.

---

1. Prepare new virtual environment with Python == 3.8.
2. Install the plugin

```
pip install "kedro-snowflake>=0.1.2"
```

3. Create new project from our starter

```
kedro new --starter=snowflights --checkout=0.1.2

 Project Name
 ============
 Please enter a human readable name for your new project.
 Spaces, hyphens, and underscores are allowed.
  [Snowflights]:

 Snowflake Account
 =================
 Please enter the name of your Snowflake account.
 This is the part of the URL before .snowflakecomputing.com
  []: abc-123
```

(continues on next page)

(continued from previous page)

```
Snowflake User
==============
Please enter the name of your Snowflake user.
 []: user2137

Snowflake Warehouse
===================
Please enter the name of your Snowflake warehouse.
 []: compute-wh

Snowflake Database
==================
Please enter the name of your Snowflake database.
 [DEMO]:

Snowflake Schema
================
Please enter the name of your Snowflake schema.
 [DEMO]:

Snowflake Password Environment Variable
=======================================
Please enter the name of the environment variable that contains your Snowflake password.
Alternatively, you can re-configure the plugin later to use Kedro's credentials.yml
 [SNOWFLAKE_PASSWORD]:

Pipeline Name Used As A Snowflake Task Prefix
=============================================

 [default]:

Enable Mlflow Integration (See Documentation For The Configuration Instructions)
================================================================================

 [False]:


The project name 'Snowflights' has been applied to:
- The project title in /tmp/snowflights/README.md
- The folder created for your project in /tmp/snowflights
- The project's python package in /tmp/snowflights/src/snowflights
```

Pipeline name parameter is here to allow you run many pipelines in the same database in snowflake and avoid conflicts between them. For demo it's fine to leave it as default.

Leave the mlflow integration disabled for now. More instructions on how to get the integration to work will available later in a blog post.

4. The `Snowflake Password Environment Variable` is the name of the environment variable that contains your Snowflake password. Make sure to set in in your current terminal session. Alternatively, you can re-configure the plugin later to use Kedro's credentials.yml. For example (using env var):

```
export SNOWFLAKE_PASSWORD="super_secret!"
```

5. Go to the project's directory: `cd snowflights`
6. Install the requirements

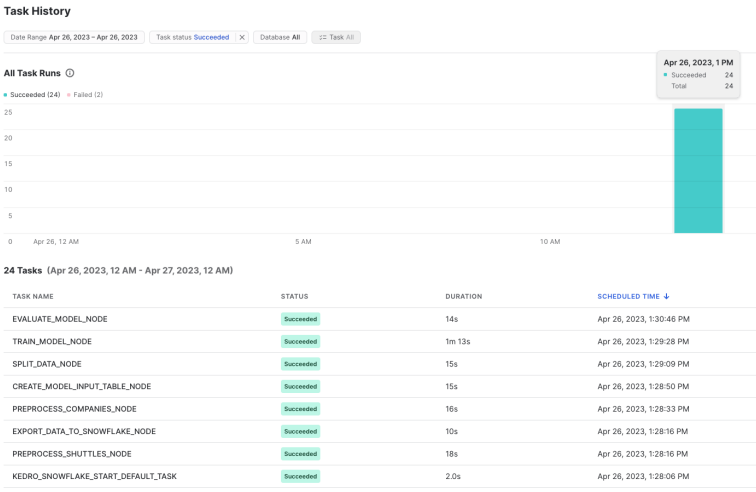
```
pip install -r src/requirements.txt
```

7. Launch Kedro pipeline in Snowflake

```
kedro snowflake run --wait-for-completion
```

After launching the command, you will see auto-refreshing CLI interface, showing the progress of the tasks execution.

In Snowpark, you can also see the history of the tasks execution:



CHAPTER

FOUR

ADVANCED CONFIGURATION

This plugin uses **snowflake.yml* configuration file in standard Kedro's config directory to handle all its configuration. Follow the comments in the example config, to understand the meaning of each field and modify them as you see fit.

```
snowflake:
  connection:
    # Either credentials name (Reference to a key in credentials.yml as in standard␣
↪Kedro)
    # or leave
    # credentials: ~
    # and specify rest of the fields
    credentials: snowflake
#    account: "abc-123"
#    database: "KEDRO"
    # Name of the environment variable to take the Snowflake password from
#    password_from_env: "SNOWFLAKE_PASSWORD"
#    role: ~
#    schema: "PUBLIC"
#    user: "user2137"
#    warehouse: "DEFAULT"
  runtime:
    # Default schedule for Kedro tasks
    schedule: "11520 minute"

    # Optional suffix for all kedro stored procedures
    stored_procedure_name_suffix: ""

    # Names of the stages
    # `stage` is for stored procedures etc.
    # `temporary_stage` is for temporary data serialization
    stage: "@KEDRO_SNOWFLAKE_STAGE"
    temporary_stage: '@KEDRO_SNOWFLAKE_TEMP_DATA_STAGE'

    # List of Python packages and imports to be used by the project
    # We recommend that this list will be add-only, and not modified
    # as it may break the project once deployed to Snowflake.
    # Modify at your own risk!
    dependencies:
      # imports will be taken from local environment and will get uploaded to Snowflake
      imports:
      - kedro
```

(continues on next page)

(continued from previous page)

```
    - kedro_datasets
    - kedro_snowflake
    - omegaconf
    - antlr4
    - dynaconf
    - anyconfig
    # packages use official Snowflake's Conda Channel
    # https://repo.anaconda.com/pkgs/snowflake/
    packages:
    - snowflake-snowpark-python
    - cachetools
    - pluggy
    - PyYAML==6.0
    - jmespath
    - click
    - importlib_resources
    - toml
    - rich
    - pathlib
    - fsspec
    - scikit-learn
    - pandas
    - zstandard
    - more-itertools
    - openpyxl
    - backoff
  # Optionally provide mapping for user-friendly pipeline names
  pipeline_name_mapping:
   __default__: default
```

CHAPTER

FIVE

SNOWFLAKE DATASETS

This plugin integrates with Kedro's datasets and provides additional set of datasets for Snowflake. The `catalog.yml` in our official Snowflights starter shows example usage of each of them:

```yaml
companies:
  type: kedro_datasets.snowflake.SnowparkTableDataSet
  table_name: companies
  database: kedro
  schema: PUBLIC
  credentials: snowflake

reviews:
  type: pandas.CSVDataSet
  filepath: data/01_raw/reviews.csv

shuttles:
  type: pandas.ExcelDataSet
  filepath: data/01_raw/shuttles.xlsx
  load_args:
    engine: openpyxl # Use modern Excel engine, it is the default since Kedro 0.18.0

preprocessed_shuttles:
  type: kedro_snowflake.datasets.native.SnowflakeStageFileDataSet
  stage: "@KEDRO_SNOWFLAKE_TEMP_DATA_STAGE"
  filepath: data/02_intermediate/preprocessed_shuttles.csv
  credentials: snowflake
  dataset:
    type: pandas.CSVDataSet
```

CHAPTER

SIX

KEDRO SNOWFLAKE DATA CLASSES

`kedro-snowflake` natively supports Kedro's official `SnowparkTableDataSet` and adds a few new classes to make it easier to use Snowflake with Kedro.

Both of these can be found under the `kedro_snowflake.datasets.native` module.

For details on usage, see the *API Reference* below.

6.1 API Reference

class kedro_snowflake.datasets.native.**SnowflakeStageFileDataSet**(*stage: str*, *filepath: str*, *dataset: str | dict*, *filepath_arg: str = 'filepath'*, *database: str | None = None*, *schema: str | None = None*, *credentials: Dict[str, Any] | None = None*)

Dataset providing an integration with *most* of the standard Kedro file-based datasets. It allows to store/load data from Snowflake stage for any underlying dataset, e.g. pandas.CSVDataSet etc.

6.1.1 Args

- `stage`: Name of the Snowflake stage. Must start with `@`.
- `filepath`: Path to the file in the Snowflake stage.
- `dataset`: a dictionary for configuring the underlying dataset.
  It can be either a string with only dataset name (e.g. "pandas.CSVDataSet")
  or a dictionary with the same structure as you would use in the Kedro catalog.yml.
- `filepath_arg`: Name of the argument in the underlying dataset that accepts the filepath (default is *filepath*). # noqa
- `database`: Name of the Snowflake database. If not specified, will attempt to load from the credentials.
- `schema`: Name of the Snowflake schema. If not specified, will attempt to load from the credentials.
- `credentials`: Credentials to use to load/save data from Snowflake. Can be used instead of *schema*/*database* # noqa in the same fashion as in the `kedro_datasets.snowflake.snowpark_dataset.SnowparkTableDataSet`.

6.1.2 Example

Example of a catalog.yml entry:

```
preprocessed_shuttles:
  type: kedro_snowflake.datasets.native.SnowflakeStageFileDataSet
  stage: "@KEDRO_SNOWFLAKE_TEMP_DATA_STAGE"
  filepath: data/02_intermediate/preprocessed_shuttles.csv
  credentials: snowflake
  dataset:
    type: pandas.CSVDataSet
```

CHAPTER

SEVEN

DEVELOPMENT

7.1 Prerequisites

- poetry `1.3.2` (as of 2023-04-26)
- Python == 3.8
- Snowflake account (or trial)

7.2 Local development

It's easiest to develop the plugin by having a side project created with Kedro (e.g. our Snowflights starter), managed by Poetry (since there is no `pip install -e` support in Poetry). In the side project, just add the following entry in `pyproject.toml`:

```
[tool.poetry.dependencies]
kedro-snowflake = { path = "<full path to the plugin on local machine>", develop = true}
```

and invoke

```
poetry update
poetry install
```

and all of the changes made in the plugin will be immediately visible in the side project (just as with `pip install -e` option).

From that point you can just use

```
kedro snowflake run --wait-for-completion
```

to start the pipelines in Snowflake and develop new features/fix bugs.

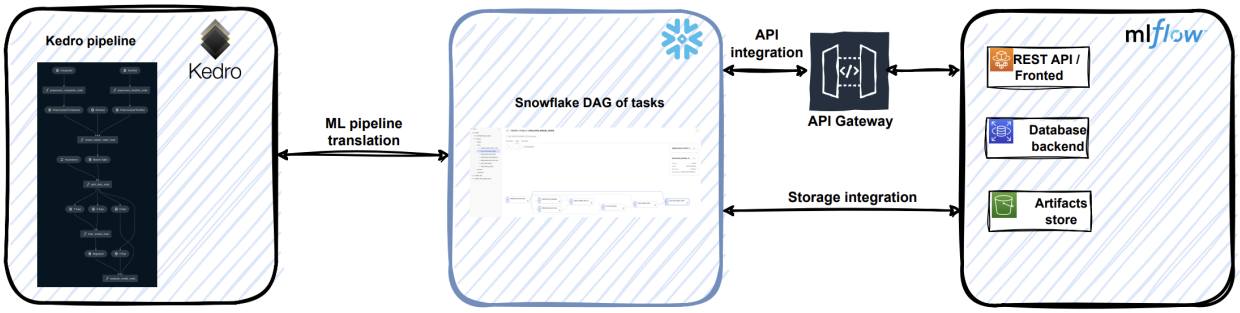
CHAPTER

EIGHT

[BETA] MLFLOW SUPPORT

8.1 High level architecture

The key challenge is to provide access to the external service endpoints (like MLflow) that is currently not yet supported natively in Snowpark (External Access feature is on the Snowflake roadmap). Snowflake external functions are the preferred workaround.



MLflow and Kedro-snowflake

8.2 Implementation details

Kedro-Snowflake <-> MLflow integration is based on the following concepts:

- Snowflake external functions that are used for wrapping POST requests to the MLflow instance. In the minimal setup the following wrapping external functions for MLflow REST API calls must be created:
  - Create run
  - Update run
  - Log param
  - Log metric
  - Search experiment
- Snowflake externa function translators for changing the format of the data sent/received from the MLflow instance.
- Snowflake API integration for setting up a communication channel from the Snowflake instance to the cloud HTTPS proxy/gateway service where your MLflow instance is hosted (e.g. Amazon API Gateway, Google Cloud API Gateway or Azure API Management).

- Snowflake storage integration to enable your Snowflake instance to upload artifacts (e.g. serialized models) to the cloud storage (Amazon S3, Azure Blob Storage, Google Cloud Storage) used by the MLflow instance.

8.3 Configuration example

```
mlflow:
  # MLflow experiment name for tracking runs
  experiment_name: demo-mlops
  stage: "@MLFLOW_STAGE"
  # Snowflake external functions needed for calling MLflow instance
  functions:
    experiment_get_by_name: demo.demo.mlflow_experiment_get_by_name
    run_create: demo.demo.mlflow_run_create
    run_update: demo.demo.mlflow_run_update
    run_log_metric: demo.demo.mlflow_run_log_metric
    run_log_parameter: demo.demo.mlflow_run_log_parameter
```

8.4 Kedro starter

The provided Kedro starter (Snowflights) has a builtin MLflow support. You can enable it during the project setup, i.e.:

```
   Enable Mlflow Integration (See Documentation For The Configuration Instructions)
   ================================================================================

    [False]: True
```

8.5 Deployment to Snowflake and inference

You can find instructions on how to make `mlflow-snowflake` integration here: https://github.com/Snowflake-Labs/mlflow-snowflake

8.5.1 Deployment

8.5.2 Inference with User Defined Function (UDF)

```
select
    MLFLOW$SNOWFLIGHTS_MODEL(
        "engines",
        "passenger_capacity",
        "crew",
        "d_check_complete",
        "moon_clearance_complete",
        "iata_approved",
        "company_rating",
        "review_scores_rating"
    ) AS price
```

(continues on next page)

(continued from previous page)

```
from
    (
        select
            1 as "engines",
            100 as "passenger_capacity",
            5 as "crew",
            true as "d_check_complete",
            true as "moon_clearance_complete",
            true as "iata_approved",
            10.0 as "company_rating",
            5.0 as "review_scores_rating"
        union all
        select
            2,
            20,
            5,
            false,
            false,
            false,
            3.0,
            5.0
    );
```

CHAPTER

NINE

INDICES AND TABLES

- genindex
- modindex
- search

INDEX

S