Snowflake-Labs / snowcli Public

44 stars 12 forks

Star Watch

Code Issues 17 Pull requests Actions Projects Security Insights

main

sfc-gh-jhollan ... 3 days ago

View code

Snowflake Developer CLI

⚠️ This is an early concept CLI for working with Snowflake. It is not officially supported by Snowflake, and is not intended for production use. ⚠️

README.md

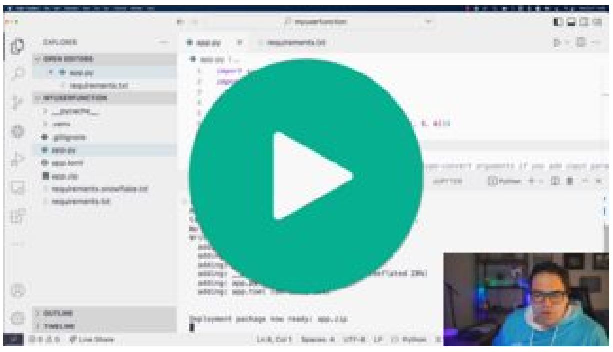
Overview

SnowCLI is a command line interface for working with Snowflake. It lets you create, manage, update, and view apps running in Snowflake.

This is an open source project and contributions are welcome (though the project is maintained on a best-effort basis).

We plan to incorporate some patterns and features of this CLI into the Snowflake CLI (SnowSQL) in the future. We hope this project starts a conversation about what a delightful developer experience could look like, and we'd love your help in shaping that with us!

Tour and quickstart



Benefits of SnowCLI

SnowCLI lets you locally run and debug Snowflake apps, and has the following benefits:

- Has support for Snowpark Python **user defined functions** and **stored procedures**, **warehouses**, and **Streamlit** apps.
- Define packages using `requirements.txt` , with dependencies automatically added via integration with Anaconda at deploy time.
- Use packages in `requirements.txt` that aren't yet in Anaconda and have them manually included in the application package deployed to Snowflake (only works with packages that don't rely on native libraries).
- Update existing applications with code and dependencies automatically altered as needed.
- Deployment artifacts are automatically managed and uploaded to Snowflake stages.

Limitations of SnowCLI

SnowCLI has the following limitations:

- You must have the SnowSQL configuration file to authenticate to SnowCLI. See the Prerequisites for more details.
- Authentication and connections are not cached between calls, so if you use authentication `externalbrowser` you must authenticate every command via the browser. See issue #19.
- Support for Windows is not yet validated. SnowCLI has been primarily tested on MacOS and Linux.
- To run Streamlit in Snowflake using SnowCLI, your Snowflake account must have access to the Streamlit private preview.

Install SnowCLI

Install with Homebrew (Mac only)

Requires Homebrew.

```
brew tap sfc-gh-jhollan/snowcli
brew install snowcli
snow --help
```

Install with pip (PyPi)

Requires Python >= 3.8

```
pip install snowflake-cli-labs
snow --help
```

Install from source

Requires Python >= 3.10 and git

```
git clone https://github.com/snowflake-labs/snowcli
cd snowcli
# you can also do the below in an active virtual environment:
# python -m venv .venv
# source .venv/bin/activate
pip install -r requirements.txt
hatch build && pip install .
snow --version
```

You should now be able to run `snow` and get the CLI message.

Get started using SnowCLI

Use SnowCLI to build a function or stored procedure, or create a streamlit if you have access to the Streamlit in Snowflake private preview.

Prerequisites

You must add your credentials to connect to Snowflake before you can use SnowCLI.

Add credentials with SnowCLI

SnowCLI uses the same configuration file as SnowSQL. If you don't have SnowSQL installed, download SnowSQL and install it. See Installing SnowSQL. You do not need to set up SnowSQL to use SnowCLI.

After installing SnowSQL, use the following SnowCLI command to add your Snowflake account credentials:

`snow connection add`

Provide a name for your connection, your account, username, and password for Snowflake.

Add credentials manually on *nix systems

If you do not want to install SnowSQL, you can add Snowflake account credentials manually:

1. In your home directory, make `.snowsql` directory with a `config` file:

   `mkdir .snowsql cd .snowsql touch config`

2. Open the config file for editing:

   `vi config`

3. Add a new configuration for your Snowflake connection with SnowCLI. You must prefix the configuration with `connections.` .

For example, to add a Snowflake account `myaccount` for a user profile `johndoe` and a password of `hunter2` , add the following:

```
[connections.connection_name]
accountname = myaccount
username = jondoe
password = hunter2
```

Add credentials manually using Windows

1. Create a `.snowsql` folder with a `config` file at the following path: `%USERPROFILE%\.snowsql\config`
2. Add a new configuration for your Snowflake connection with SnowCLI. You must prefix the configuration with `connections.` .

Build a function

To build a function or a stored procedure using SnowCLI, do the following:

1. Navigate to an empty directory to create your function.

2. Run the command:

   ```
   snow function init
   ```

   SnowCLI populates the directory with the files for a basic function. You can open `app.py` to see the files.

3. Test the code by running the `app.py` script:

   ```
   python app.py
   ```

   You see the message: `Hello World!`

4. Package the function:

   ```
   snow function package
   ```

   This creates an `app.zip` file that has your files in it

5. Log in to snowflake:

   ```
   snow login
   ```

6. Configure your first environment:

   ```
   snow configure
   ```

7. Create a function:

   ```
   snow function create
   ```

8. Try running the function:

   ```
   snow function execute -f 'helloFunction()'
   ```

   You see Snowflake return the message: 'Hello World!'

After building your function, you can modify `app.py` , `requirements.txt` , or other files and follow a similar flow, or update a function by running the following:

```
snow function update -n <myfunction> -f <app.zip>
```

Create a Streamlit

To create a Streamlit, do the following:

Note: Your account must have access to the Streamlit in Snowflake private preview to create a Streamlit.

1. Change to a directory with an existing Streamlit app, or create a directory for a new Streamlit app.

2. Log into Snowflake::

   ```
   snow login
   ```

3. Create an environment and select your database, schema, role, and warehouse:

   ```
   snow configure
   ```

   The environment name defaults to 'dev'.

4. Create a streamlit with a name that you specify:

   ```
   snow streamlit create <name>
   ```

   If you don't specify a name, the file defaults to `streamlit_app.py` .

5. Deploy your app and open it in the browser:

   ```
   snow streamlit deploy <name> -o
   ```

Create a stored procedure

See Build a function.

Contributing

If interested in contributing, you will want to instanstiate the pre-commit logic to help with formatting and linting of commits. To do this, run the following in the `snowcli` cloned folder on your development machine:

```
pip install pre-commit
pre-commit
```

Get involved

Have a feature idea? Running into a bug? Want to contribute? We'd love to hear from you! Please open or review issues, open pull requests, or reach out to us on Twitter or LinkedIn @jeffhollan and @jroes.

Releases 19

v0.1.18 Latest
3 days ago

+ 18 releases

Packages

No packages published

Used by 1

@sfc-gh-jhansen / **tko-data-engineering**

Contributors 11

Languages

● **Python** 100.0%