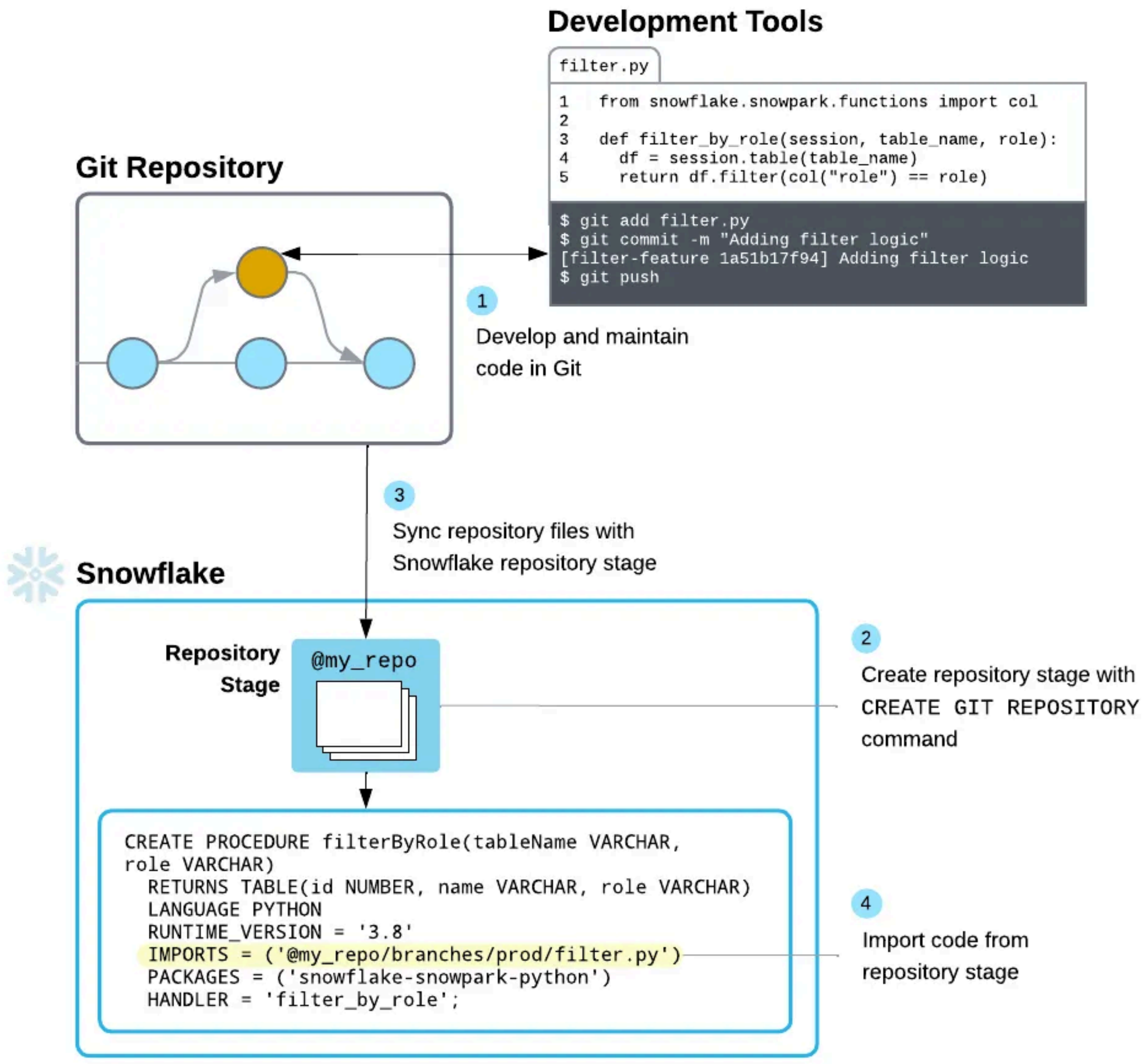
Connect Git to Snowflake, now in Public Preview

Jeff Hollan · Follow
Published in Snowflake · 4 min read · 1 day ago

We're excited to announce that the first wave of features for integrating git repos and git workflows into Snowflake is now in Public Preview across all clouds! With Snowflake git integration, you can connect securely to a git repo hosted outside of Snowflake (GitHub, GitLab, Azure DevOps, or BitBucket currently supported) and have the contents of that repo synced to your Snowflake account. This allows you to have the "source of truth" for SQL scripts, Snowpark functions and procedures, Native Apps, and Streamlit Apps to pull from git.


Development Tools
filter.py
Git Repository
1
Develop and maintain code in Git
3
Sync repository files with Snowflake repository stage
Snowflake
Repository Stage
@my_repo
2
Create repository stage with CREATE GIT REPOSITORY command
4
Import code from repository stage

```
from snowflake.snowpark.functions import col

def filter_by_role(session, table_name, role):
  df = session.table(table_name)
  return df.filter(col("role") == role)
```

```
$ git add filter.py
$ git commit -m "Adding filter logic"
[filter-feature 1a51b17f94] Adding filter logic
$ git push
```

```
CREATE PROCEDURE filterByRole(tableName VARCHAR,
role VARCHAR)
  RETURNS TABLE(id NUMBER, name VARCHAR, role VARCHAR)
  LANGUAGE PYTHON
  RUNTIME_VERSION = '3.8'
  IMPORTS = ('@my_repo/branches/prod/filter.py')
  PACKAGES = ('snowflake-snowpark-python')
  HANDLER = 'filter_by_role';
```

Diagram of a “git flow” within Snowflake

For this initial public preview, you can only access and read files from your git repo and not alter or commit those files back into the git repo (this will be coming in the future — stay tuned!). You can use any tool that integrates with git for writing or collaborating on changes in the meantime, such as VS Code.

Setting up a connection to a git repo

Consider a git repo like this one that has a number of files I want to be able to execute in my Snowflake account.

First I need to connect this git repo to my Snowflake account. I need to create a few things:
API Integration—an ACCOUNTADMIN must create this, which allows git traffic from my Snowflake account to the URL or URL patterns specified.

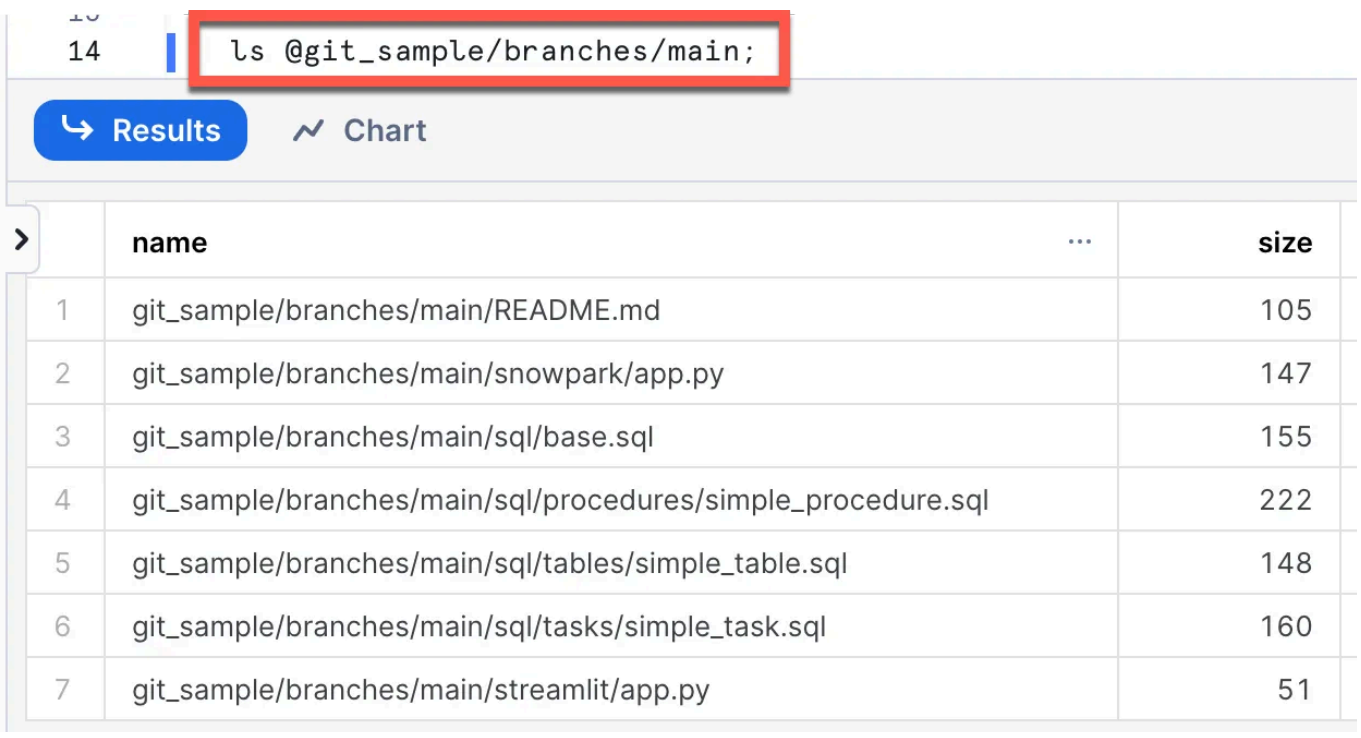
```
CREATE OR REPLACE API INTEGRATION git_sample_integration
  API_PROVIDER = git_https_api
  API_ALLOWED_PREFIXES = ('https://github.com/jeffhollan')
  ENABLED = TRUE;
```

Note: If this was a private repo, I could also create a SECRET with the authentication information (a personal access token scoped to the appropriate repo(s)) that could be used to authenticate.

Git Repository — this is the Snowflake representation of the external git repo and includes a cache of all files from all branches / tags / commits.

```
CREATE OR REPLACE GIT REPOSITORY git_sample
  API_INTEGRATION = git_sample_integration
  -- GIT_CREDENTIALS = my_secret if needed
  ORIGIN = 'https://github.com/jeffhollan/snowflake-git-sample';
```

Now in the database/schema created, I have a special kind of stage, a `git repository` , which has all files from the git repo. It has a special logical naming structure so I can navigate across files in different branches. For example, notice the output of the following command mirrors the main branch of my repo in GitHub:



```
ls @git_sample/branches/main;
```

	name	size
1	git_sample/branches/main/README.md	105
2	git_sample/branches/main/snowpark/app.py	147
3	git_sample/branches/main/sql/base.sql	155
4	git_sample/branches/main/sql/procedures/simple_procedure.sql	222
5	git_sample/branches/main/sql/tables/simple_table.sql	148
6	git_sample/branches/main/sql/tasks/simple_task.sql	160
7	git_sample/branches/main/streamlit/app.py	51

Use case #1: SQL Scripts and deployments backed by git

One instrumental pattern with git integration is using git as the store for SQL scripts or tasks. With git, you can have governed source control and better controls over the content in those files.

You'll notice in the repo above I have a number of SQL files I want to be able to store in git and execute in Snowflake. Now that I have created the Git Repository object, I can execute SQL files using EXECUTE IMMEDIATE FROM.

In my repo I have a file that creates a table (if not exists), creates a procedure (or replaces), and creates a task (if not exists). I also have a `base.sql` file which calls the other files in the right order. I can execute all of these files in

Snowflake which will execute the latest version of them in my `main` branch with the following:

```
EXECUTE IMMEDIATE FROM @git_sample/branches/main/sql/base.sql;
```

This will then execute all the files as they exist in the Git Repository and create the necessary objects if they don't exist.

(It's worth calling out some features coming soon to allow you to CREATE OR ALTER and add Jinja templating to SQL files really take this feature to the next level!)

I can then update the SQL files in GitHub, fetch the git repo to get the latest changes, and re-execute any files as needed.

Use case #2: Snowpark Python backed by git

Notice in my sample GitHub repo I have a simple Snowpark procedure. I can create a procedure in my Snowflake account that imports this file and uses it as the source for the procedure. What's nice about this pattern is that as I push changes to this file / branch and `fetch` those changes in Snowflake, my procedure will automatically update.

Notice the **IMPORTS** statement in the create below to see how I reference this file.

```
CREATE OR REPLACE PROCEDURE hello()
RETURNS TABLE()
```

```
LANGUAGE PYTHON
RUNTIME_VERSION='3.11'
PACKAGES=('snowflake-snowpark-python')
IMPORTS=('@git_sample/branches/main/snowpark/app.py')
HANDLER='app.return_simple_table';
```

You could imagine I could have different copies of this procedure in different schemas, each mapping to a different branch (e.g. my PROD schema points to `main` , while my DEV schema points to `dev`).

Use case #3: Streamlit backed by git

Similar to the above, I can have Streamlit apps that pull from Git Repository objects.

```
CREATE STREAMLIT git_sample
ROOT_LOCATION='@git_sample/branches/main/streamlit'
MAIN_FILE='app.py';
```

NOTE: In this mode, you will be able to view Streamlits in Snowflake, but the edit experience will not work and will throw exceptions. You can still update the Streamlit by updating the file contents in the external git repo.

Git started with git today ;)

This isn't an exhaustive list of scenarios. For instance, you can use the git integration for a Native Apps package. But this should give you a good sense of how to start using the git integration today. As mentioned throughout the blog, this is just the first phase of deeper integration of git + Snowflake. Keep