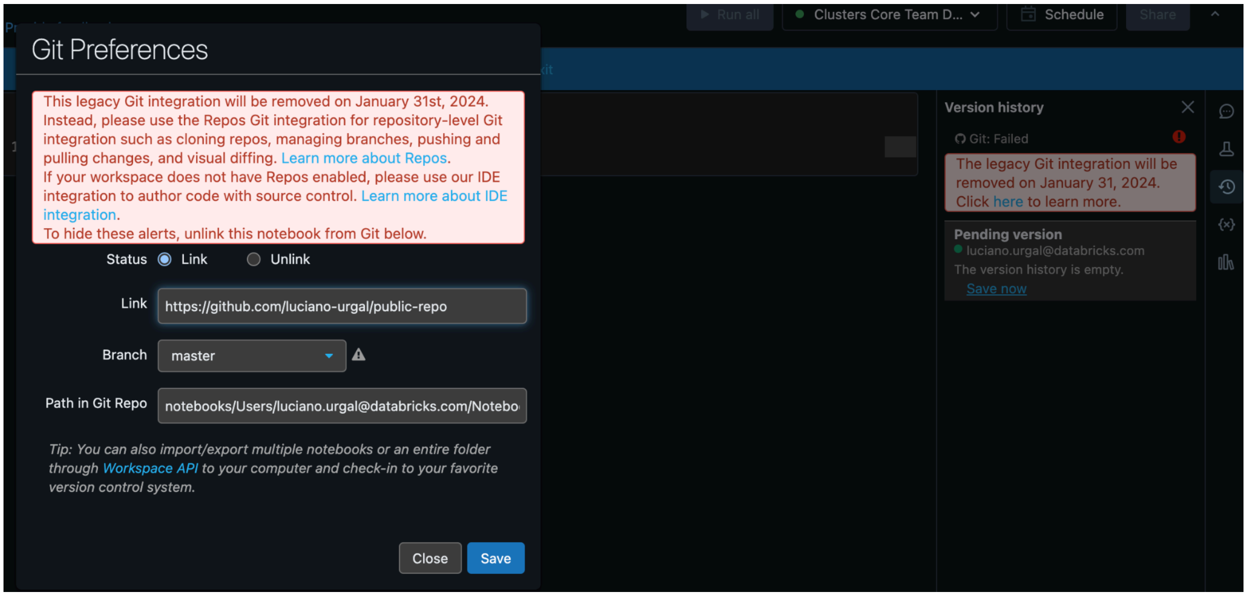
Switching to Databricks Repos from Legacy Git integration

Databricks Repos effectively behave as embedded Git clients in your workspace. Unlike the legacy Git integration, which only works on a per-notebook level, Databricks Repos allows you to perform **Git operations for multiple notebooks and files** with the same Git connection. The legacy Git integration has been in legacy status for over two years, and the deprecation notice has been displayed in the UI since Nov 3rd, 2023.



Git Preferences

This legacy Git integration will be removed on January 31st, 2024. Instead, please use the Repos Git integration for repository-level Git integration such as cloning repos, managing branches, pushing and pulling changes, and visual diffing. Learn more about Repos.
If your workspace does not have Repos enabled, please use our IDE integration to author code with source control. Learn more about IDE integration.
To hide these alerts, unlink this notebook from Git below.

Status ● Link ○ Unlink

Link: https://github.com/luciano-urgal/public-repo

Branch: master

Path in Git Repo: notebooks/Users/luciano.urgal@databricks.com/Notebo

Tip: You can also import/export multiple notebooks or an entire folder through Workspace API to your computer and check-in to your favorite version control system.

Version history

Git: Failed

The legacy Git integration will be removed on January 31, 2024. Click here to learn more.

Pending version
luciano.urgal@databricks.com
The version history is empty.
Save now

To set up a Databricks Repo:

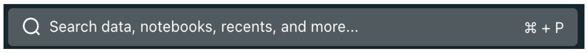
1. Get the Git URL and provider from your existing legacy Git integration.
2. Clone the remote repo: This step will automatically use the same credentials you configured for the legacy Git integration.
    a. Optionally, if you have a large remote repo and only need a subset of directories, you can use the sparse checkout mode to limit the directories you clone.
3. Voila! You now have notebooks connected to your remote repository via Databricks Repos.

You can find information on committing changes, pulling changes, and doing other Git operations with Repos in our documentation. To preserve uncommitted changes in a notebook

- Clone the remote repo with this notebook into a Databricks Repo
- Move this notebook to the repo into the same location to replace the old version
- Commit and push

As a best practice, make team collaboration more effective by **creating a Databricks Repo for each user that integrates with the same Git repository**. If each user works in their own development branch, each user should also set up and use their own Repo pointing to that branch. Although multiple users can edit contents in a Repo, only one designated user should perform Git operations (such as pull, push, commit, and branch switching) for that Repo. Databricks strongly recommends against multiple users performing Git operations in the same Repo as this can result in errors and inconsistent results.

After creating a new Databricks Repo, update any Workflows, `dbutils.notebook.run`, `%run`, or other references that refer to the notebooks by workspace paths.

- **Notebook references**: Search for the notebook path and name via the global search bar inside the Workspace .
- **Workflow references**: Use the Databricks SDK, API, or CLI to list jobs with expanded task details.
- **External orchestrator references**: Update references from data orchestrators such as Airflow or Azure Data Factory.