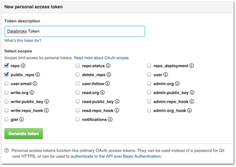
GitHub Version Control

This topic describes how to set up version control for notebooks using GitHub through the UI. Although this document describes how to set up GitHub integration through the UI, you can also integrate through the [Databricks CLI](https://docs.databricks.com/user-guide/dev-tools/databricks-cli.html#databricks-cli) or [Workspace API](https://docs.databricks.com/api/latest/workspace.html#workspace-api).

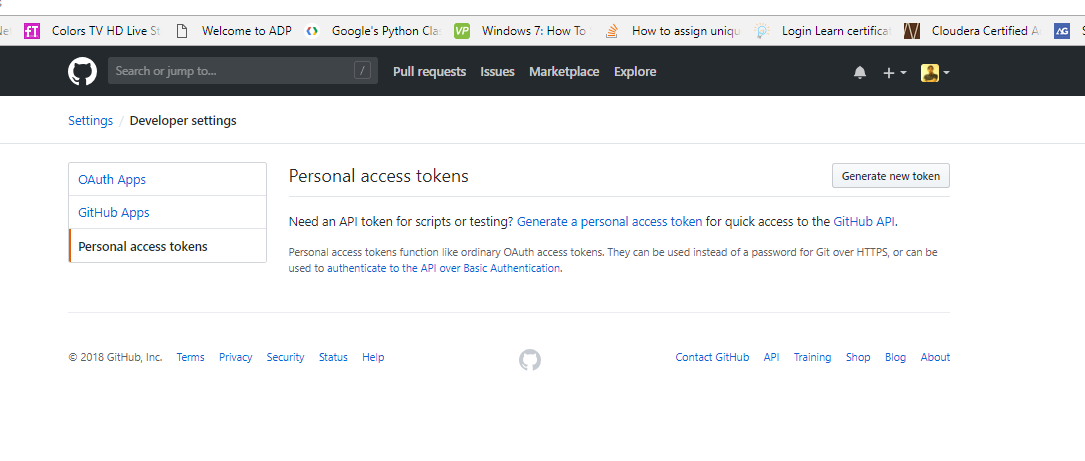
## [**Configure version control**](https://docs.databricks.com/user-guide/notebooks/github-version-control.html#id1)

Configuring version control involves creating access credentials in your version control provider and adding those credentials to Databricks.

1. From GitHub, access the menu on the upper right, next to your Gravitar, and select **Settings**.
2. Go to the **Personal access tokens** tab.
3. Click the **Generate New Token** button to create a new token.
4. Select the **repo** and **public\_repo** permissions, and click the **Generate Token** button.

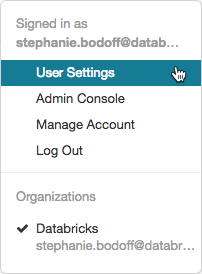


1. Copy the token to your clipboard. You enter this token in Databricks in the next step.

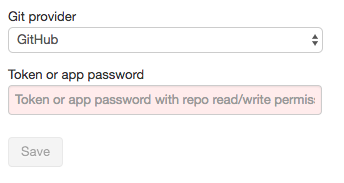


### [**Save your access token to Databricks**](https://docs.databricks.com/user-guide/notebooks/github-version-control.html#id3)

1. Click the User icon Account Icon at the top right of your screen and select **User Settings**.



1. Click the **Git Integration** tab.
2. If you have previously entered credentials, click the **Change token or app password** button.
3. In the Git provider drop-down, select **GitHub**.



1. Paste your token into the **Token or app password** field and click **Save**.

## [**Work with notebook revisions**](https://docs.databricks.com/user-guide/notebooks/github-version-control.html#id4)

You work with notebook revisions in the History panel. Open the history panel by clicking **Revision history** at the top right of the notebook.

../../_images/revision-history-open.png

**Note**

You cannot modify a notebook while the History panel is open.

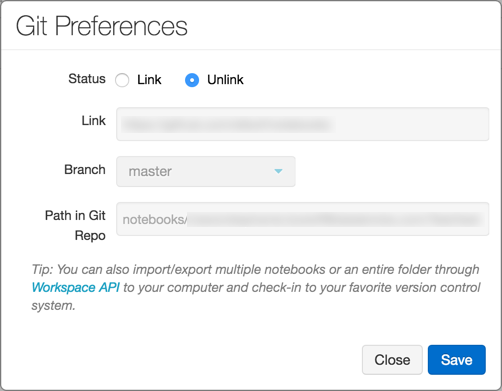
### [**Link a notebook to GitHub**](https://docs.databricks.com/user-guide/notebooks/github-version-control.html#id5)

1. Open the History panel. The Git status bar displays **Git: Not linked**.

../../_images/git-not-linked.png

1. Click **Git: Not linked**.

The Git Preferences dialog displays. The first time you open your notebook, the Status is **Unlink**, because the notebook is not in GitHub.



1. In the Status field, click **Link**.
2. In the Link field, paste the URL of the GitHub repository.
3. Click the **Branch** drop-down and select a branch or type the name of a new branch.
4. In the Path in Git Repo field, specify where in the repository to store your file.

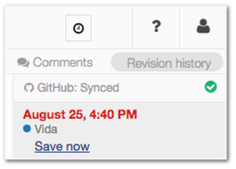
Python notebooks have the suggested default file extension .py. If you use .ipynb, your notebook will save in iPython notebook format. If the file already exists on GitHub, you can directly copy and paste the URL of the file.

1. Click **Save** to finish linking your notebook. If this file did not previously exist, a prompt with the option **Save this file to your GitHub repo** displays.
2. Type a message and click **Save**.

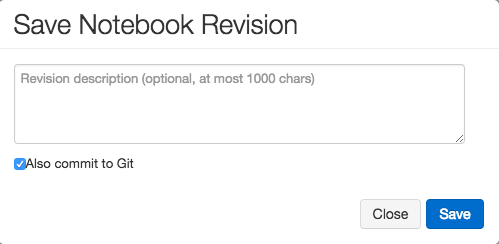
### [**Save a notebook to GitHub**](https://docs.databricks.com/user-guide/notebooks/github-version-control.html#id6)

While the changes that you make to your notebook are saved automatically to the Databricks revision history, changes do not automatically persist to GitHub.

1. Open the History panel.



1. Click **Save Now** to save your notebook to GitHub. The Save Notebook Revision dialog displays.
2. Optionally, enter a message to describe your change.
3. Make sure that **Also commit to Git** is selected.

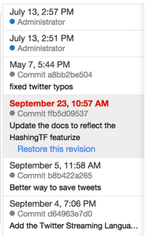
[](https://docs.databricks.com/_images/save-revision.png)

1. Click **Save**.

### [**Revert or update a notebook to a version from GitHub**](https://docs.databricks.com/user-guide/notebooks/github-version-control.html#id7)

Once you link a notebook, Databricks syncs your history with Git every time you re-open the History panel. Versions that sync to Git have commit hashes as part of the entry.

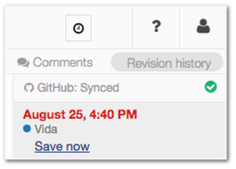
1. Open the History panel.



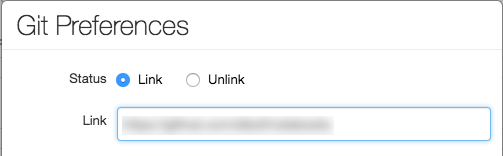
1. Choose an entry in the History panel. Databricks displays that version.
2. Click **Restore this version**.
3. Click **Confirm** to confirm that you want to restore that version.

### [**Unlink a notebook**](https://docs.databricks.com/user-guide/notebooks/github-version-control.html#id8)

1. Open the History panel.
2. The Git status bar displays **Git: Synced**.



1. Click **Git: Synced**.



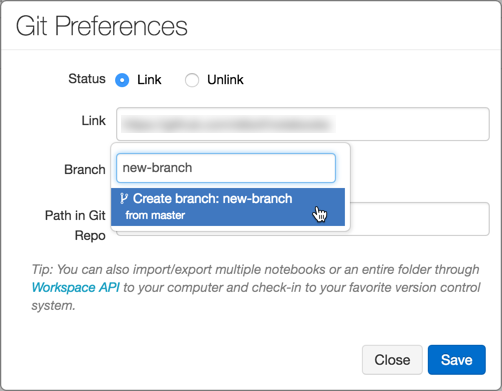
1. In the Git Preferences dialog, click **Unlink**.
2. Click **Save**.
3. Click **Confirm** to confirm that you want to unlink the notebook from version control.

### [**Branch support**](https://docs.databricks.com/user-guide/notebooks/github-version-control.html#id9)

You can work on any branch of your repository and create new branches inside Databricks.

#### **Create a branch**

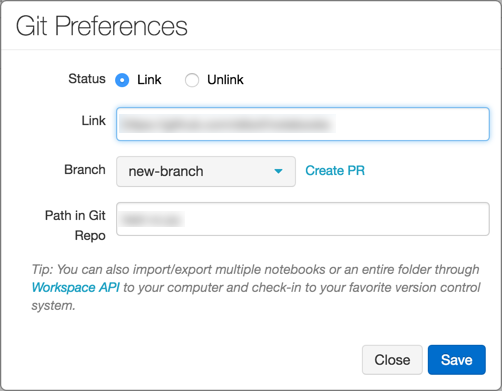
1. Open History panel.
2. Click the Git status bar to open the GitHub panel.
3. Click the **Branch** dropdown.
4. Enter a branch name.



1. Select the **Create Branch** option at the bottom of the dropdown. The parent branch is indicated. You always branch from your current selected branch.

#### **Create a pull request**

1. Open History panel.
2. Click the Git status bar to open the GitHub panel.



1. Click **Create PR**. GitHub opens to a pull request page for the branch.

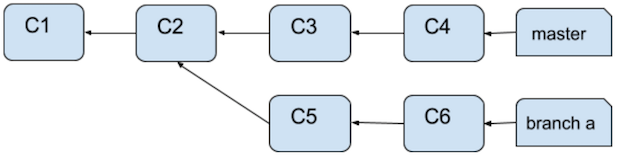
#### **Rebase a branch**

You can also rebase your branch inside Databricks. The **Rebase** link displays if new commits are available in the parent branch. Only rebasing on top of the default branch of the parent repository is supported.

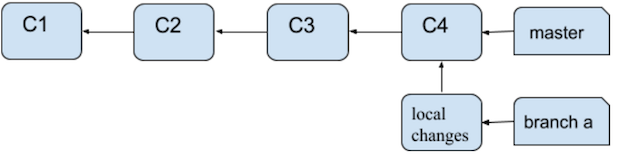
../../_images/github-rebase.png

For example, assume that you are working on databricks/reference-apps. You fork it into your own account (for example, brkyvz) and start working on a branch called my-branch. If a new update is pushed to databricks:master, then the Rebase button displays, and you will be able to pull the changes into your branch brkyvz:my-branch.

Rebasing works a little differently in Databricks. Assume the following branch structure:



After a rebase, the branch structure will look like:



What’s different here is that Commits C5 and C6 will not apply on top of C4. They will appear as local changes in your notebook. Any merge conflict will show up as follows:



You can then commit to GitHub once again using the **Save Now** button.

**What happens if someone branched off from my branch that I just rebased?**

If your branch (for example, branch-a) was the base for another branch (branch-b), and you rebase, you need not worry! Once a user also rebases branch-b, everything will work out. The best practice in this situation is to use separate branches for separate notebooks.

#### **Best practices for code reviews**

Databricks supports Git branching.

* You can link a notebook to your own fork and choose a branch.
* We recommend using separate branches for each notebook.
* Once you are happy with your changes, you can use the Create PR link in the Git Preferences dialog to take you to GitHub’s pull request page.
* The Create PR link displays only if you’re not working on the default branch of the parent repository.

## [**GitHub Enterprise**](https://docs.databricks.com/user-guide/notebooks/github-version-control.html#id10)

**Important**

Integration with GitHub Enterprise is not supported. However, you can use the [Workspace API](https://docs.databricks.com/api/latest/workspace.html#workspace-api) to programmatically create notebooks and manage the code base in GitHub Enterprise.

## [**Troubleshooting**](https://docs.databricks.com/user-guide/notebooks/github-version-control.html#id11)

If you receive errors related to syncing GitHub history, verify the following:

1. You have initialized the repository on GitHub, and it isn’t empty. Try the URL that you entered and verify that it forwards to your GitHub repository.
2. Your personal access token is active.
3. If the repository is private, you must have at least read level permissions (through GitHub) on the repository.