# GitHub Desktop for Data Scientists

**By**[**Drew Seewald**](https://realdrewdata.medium.com/)**, Data Scientist**

*Less scary than version control in the command line.*



Relax, you don’t have to use the command line

Photo by Dennis van Dalen on Unsplash

Version control is important for collaborating on code, sharing it with others, being able to view old versions of the code, and even deploying the code automatically. It can be a bit confusing at first, but is well worth your time, especially if you work in the open source space or on a team where you will frequently be using version control for projects. Here are some of the biggest features that make it worth using:

* Storing file change history with comments
* Organizing multiple users editing the same project simultaneously
* Facilitating code review procedures
* Automating workflows to report issues, request improvements, and deploy code

## Version Control Features

One of the main features of version control is the file change history for every file in the repository. This serves as a change log for each and every file, so it is always possible to see what code was running at any point in the past. Every time someone updates a file and pushes the new version to the repository, they have to add a short comment. In a perfect world, this details what the changes were and why they were made. If there is ever a question on why something was changed or why, the person responsible will be tagged in the commit, along with the additional information they provided.

Another feature of version control is the ability to create branches. A branch is a new version of the code that is kept separate. This is helpful for making and testing changes to the code as it doesn’t change the main branch, where the most up to date working version is. Branches can also be used by different users to work on different code or features at the same time. These branches can be merged back to the master when they are ready, and there is a process to reconcile the differences between them when merging.

Code review is a best practice when working with teams. One person may do all the work on a new feature in a new branch, but before blindly merging it into the main branch, it should be reviewed by the team. When a pull request is created to move code to the main branch, it also starts a discussion where team members can talk about the code and request changes before it gets merged to the main branch. This process should help improve the code that makes it into production to prevent bugs and breaks, improve the efficiency of the code, or even have it match a standard for formatting the code.

A final noteworthy benefit of using GitHub for version control is the automation options it offers. If there is a standard code review checklist, it can be added as a template that will be available when a pull request is created, ready to be filled out as review tasks are completed. Templates can also be used when people create issues so that they remember all of the details they need to input as they are creating an issue. GitHub also offers actions to support automation. These can be triggered by different events, such as merging code into the master branch. An action can run unit tests, build/compile package components, and even deploy code to production.

## Version Control Flavors

There are a few big names version control you may have heard of. Some of the most popular are Git and GitHub. [Git](https://git-scm.com/about) is the underlying technology for version control, and [GitHub](https://github.com/features) is software that simplifies the version control workflow.

Git can be used locally, without any need for an external repository. You can do all of your version control tasks on your computer’s hard drive. Local Git repositories are ideal for personal projects or for when you aren’t quite ready to share your code with your entire team, but still want the benefits of version control.

The GitHub website is a repository to store code. A lot of open source projects, such as Python and R packages, are hosted on the GitHub website. For public repositories, anyone can view revision history, issues with the package, and documentation related to it.

To connect to a repository on the GitHub website, we can use Git or GitHub Desktop. For those of you who love a command line interface (CLI), [Rebecca Vickery](https://rebecca-vickery.medium.com/) has a great article on [using the Git CLI for Data Science](https://towardsdatascience.com/introduction-to-github-for-data-scientists-2cf8b9b25fba). So why should you keep reading? **Command lines can be intimidating**. There is nothing wrong with wanting a graphical user interface (GUI) to manage your version control. GitHub Desktop provides a clear and simple interface with your repository.

## GitHub Desktop Process Flow

While everyone will have a slightly different process flow for their repository, there are a few general steps to make changes to code on GitHub:

1. Creating a branch
2. Adding commits
3. Creating a new pull request
4. Completing code review
5. Merging the pull request

Creating a branch makes a copy of the current production code. A developer will make changes to the files, committing any changes to the new branch. Next, a pull request will open discussion to add the new branch’s changes to the production code, typically in the master or main branch. Code reviewers can add comments and request clarification on the changes made on the pull request. Once the review is complete and any necessary changes made, the pull request can be merged to the master or main branch and closed.

Let’s walk through these steps in more detail, looking at how to do each one using GitHub Desktop.

## Creating a Branch (separating the new code from the old)

To make a change, first create a new branch. If you have full access to the repository, you can simply create a new branch on the repository’s GitHub site.

1a. Click on Branch: main

Interface gráfica do usuário, Texto, Aplicativo, Email, Site

Descrição gerada automaticamente

Image by the author

2a. Type the name of the new branch in the text box. There may be some branch naming conventions to consider from your organization to keep things organized

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Image by the author

3a. Click on Create branch

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Image by the author

The new branch will now be selected.

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Image by the author

If you don’t have full access to the repository, which is common for public projects, you will have to fork the repository. A new branch and a fork are synonymous. A fork will be created in a new repository as opposed to the same one as the production code, typically under your personal profile. To fork:

1b. In the top right, click on Fork

Interface gráfica do usuário, Texto, Aplicativo, chat ou mensagem de texto

Descrição gerada automaticamente

Image by the author

2b. Wait for the files to copy

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Image by the author

3b. The new fork will be selected

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Image by the author

**Adding Commits (enhancing the code/adding features)**

To make commits to the code, you will clone the repository to your local computer. This copies the code for you to work on before sending the updates back to the repository. To clone the repository to your local machine:

Click clone or download.

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Image by the author

Click open with GitHub desktop.

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Image by the author

If you don’t have GitHub Desktop, click download GitHub Desktop.

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Image by the author

GitHub desktop will ask where to clone the repository to on your local machine. This is the Local path field.

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente  
Image by the author

Click on the branch and select the newly created branch. This will update the files on your local machine with any updates on that branch and make it the active one to add commits to.

Interface gráfica do usuário, Aplicativo, Teams

Descrição gerada automaticamente  
Image by the author

To make changes, open the directory you chose when cloning and make changes with your text editor or integrated development environment (IDE) as normal. Save the files.

Return to GitHub Desktop. GitHub Desktop is constantly scanning the repository folder tree and will see any changes you make. These changes will show up on the left pane. The right pane will preview changes to the selected file (certain file types won’t preview).

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente  
Image by the author

Each time you make a set of related changes, commit those changes to your repository. Remember to add comments to the commits so that people can easily identify what was changed. The upper text box is for a quick description, but if you have more notes to add about the commit, put them in the larger description text box.

With changes made and comments added, commit the changes.

Interface gráfica do usuário, Texto, Aplicativo, chat ou mensagem de texto, Email

Descrição gerada automaticamente  
Image by the author

Committing changes only saves them on the local files. To push the changes back to the GitHub server, click push origin. If there are commits made that haven’t been pushed back to the server, a message will appear on the right pane that says **Push xx commit(s) to the origin remote**. The origin is just a name for where the repository was cloned from.

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente  
Image by the author

**Open a Pull Request**

   
Navigate to your repository on the GitHub server. Make sure you are on the correct branch. If you created a new branch on the original repository, navigate there. If you had to fork the repository, navigate to the repository on your personal profile.

On the **Pull requests** tab, click on **New pull request**.

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente  
Image by the author

Select the new branch as the one to compare, and **click create pull request**. In our case, the pull request is automatically populated with our commit comments.

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente  
Image by the author

**Code Review**

   
Code review helps make sure that the code we are adding or changing is correct and has been reviewed and approved by multiple people. Whether you have access to a repository or not, you should always have reviewers check the changes. If there are any questions, go over them as a team.

Pull requests show up under the Pull request tab of the repository. Each pull request has a conversation, commits, and files changed tabs.

Conversation is where people can add questions or comments about the code. You can format your comments and tag people and issues in your comments.

Commits shows all of the commits and comments made in the pull request

Files changed shows which files were changed, added, or deleted along with line by line comparisons of code where available

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente  
Image by the author

**Merge the Pull Request**

   
In my case, my main branch was changed while I was working on the new branch. This is why there are messages saying that there are conflicts to resolve. Clicking the Resolve conflicts button opens an editor. It will show the version of the file from each branch, allowing you to delete one and keep the other, or create some version of both.

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente  
Image by the author

In this case, the version from the main branch is correct. The changes from the other branch and the separators can be deleted. The conflict can be marked as resolved, and committed to the merge.

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente  
Image by the author

With code review and conflicts out of the way, the pull request can be merged. Once again there will be an option to give a comment on what the merge accomplishes. After merging, the code will be part of the master or main branch!

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente  
Image by the author

And there you have it, you now know how to complete the most basic version control tasks using GitHub and GitHub Desktop!

*I write about data science, analytics, and programming concepts. You can connect with me on*[*Medium*](https://realdrewdata.medium.com/lists)*,*[*Twitter*](https://twitter.com/RealDrewData)*, and*[*LinkedIn*](https://www.linkedin.com/in/realdrewdata/)*.*

**Further Reading**

   
[**Understanding the GitHub flow**](https://guides.github.com/introduction/flow/)  
When you're working on a project, you're going to have a bunch of different features or ideas in progress at any given…

[**Version Control with Git**](https://swcarpentry.github.io/git-novice/)  
Wolfman and Dracula have been hired by Universal Missions (a space services spinoff from Euphoric State University) to…

[**What is version control | Atlassian Git Tutorial**](https://www.atlassian.com/git/tutorials/what-is-version-control)  
Version control, also known as source control, is the practice of tracking and managing changes to software code…

**Bio:**[**Drew Seewald**](https://realdrewdata.medium.com/) is a Data Scientist at Mercedes-Benz Financial Services. Follow Drew on Twitter [@RealDrewData](https://twitter.com/realdrewdata?lang=en) or connect on [LinkedIn](https://www.linkedin.com/in/realdrewdata/" \t "_blank).