Maintaining Translations with GitHub Desktop

Updated: 04-Jan-2025
By: Chris Jakeman

Contents

What are *Git* and *GitHub?*

Git is a free version control system designed to track digital projects. It was created to support the development of Linux and has become the leading version control system. (Microsoft has switched from their in-house *Team Services* to *Git* and now uses it for Windows development.)

GitHub is the world's largest hosting service for software, both public and private projects. Digital assets such as Open Rails routes benefit just as much as software from this tracking capability. *GitHub* is now owned by Microsoft.

Git is a small program that you can install on your PC to keep each version of your project in a repository (or repo) where it is safe from changes. The free *GitHub Desktop* program includes a version of *Git* and also communicates between your PC and a remote repository in your account at GitHub.com. (Note that *GitHub Desktop* is for 64-bit Windows only.[1])

Git helps a translator to work independently or with partners.

[1] For 32-bit Windows, many other graphical programs are available free (such as SourceTree) and Git can also be used from the command line using Git Bash.

Maintenance Procedure

The procedure to maintain translations is:

Initially:

1. Make your own remote repository (or repo) at GitHub as a special copy (called a "fork") of the official Open Rails remote repo. Git remembers where your repo was forked from and provides a mechanism to check and adopt your changes back into the official repo.

If you are collaborating with someone else then this is the repo that you share and Git coordinates your changes so none get lost or overwritten.

2. Download a copy (call a "clone") of your remote repo to make a local repo on your PC. Once again, Git remembers where your local repo was cloned from and makes it easy to push your local changes back to your remote repo.

Thereafter:

3. Bring your repos up to date with the official repo at GitHub. Of course, you won't need to do this the very first time.

4. Make and test your changes using your translation tools.

5. Commit your changes to the local repo on your PC. This is a permanent record which keeps your work safe and so you can safely and easily reverse or repeat the changes.

6. Push your commit with all your changes back to your remote repo at GitHub.

7. Submit your commit by making a "pull request" to the official repo at GitHub.

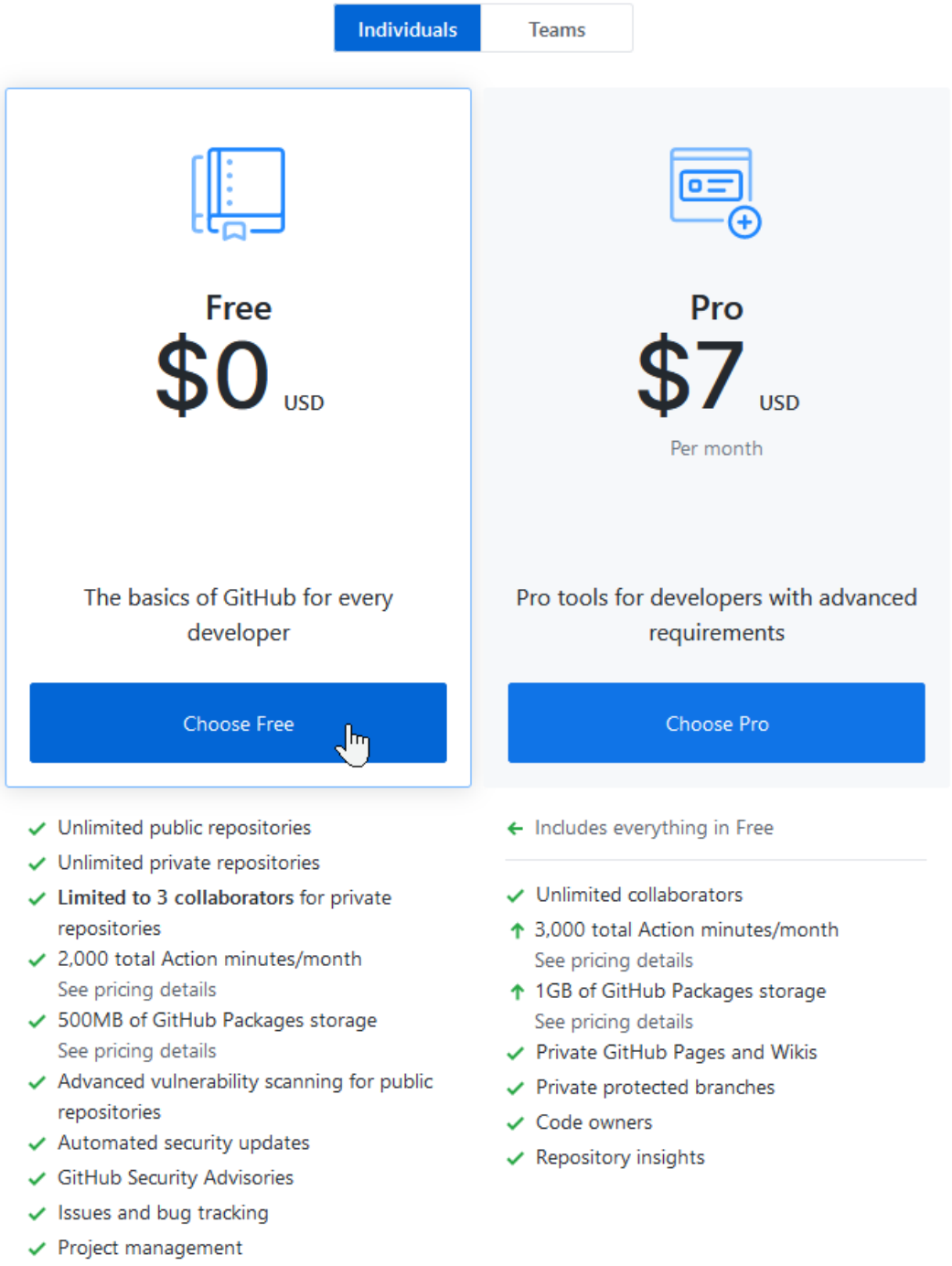
They will be checked by including them automatically in a temporary build just in case there is a problem when compiling.

The commit will then be reviewed by another member of the team and adopted by "pulling" the changes from your remote repo to the official repo. This is the reason that a submitted change is called a "pull request".

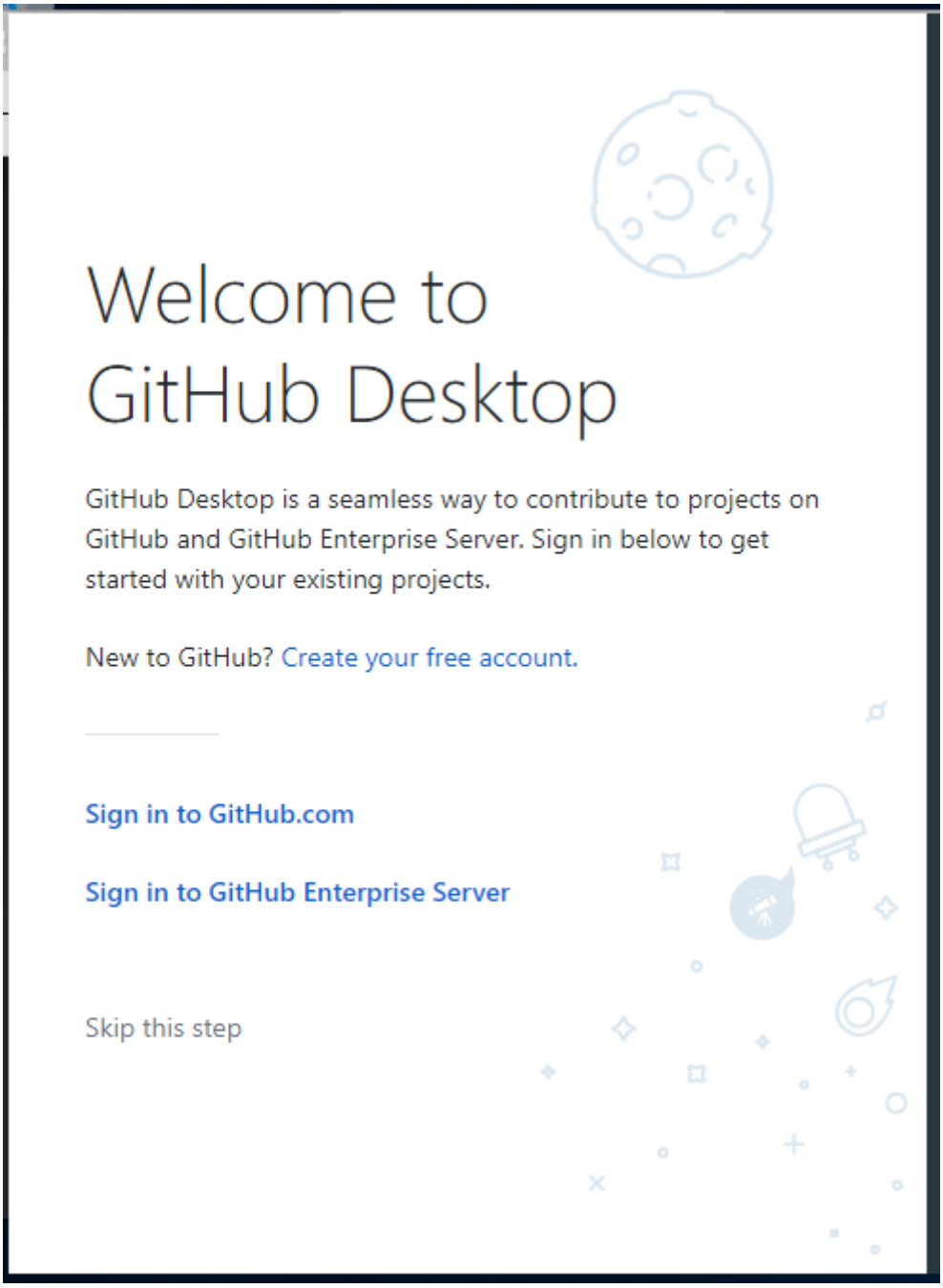
Using *GitHub Desktop*

Install *GitHub Desktop*

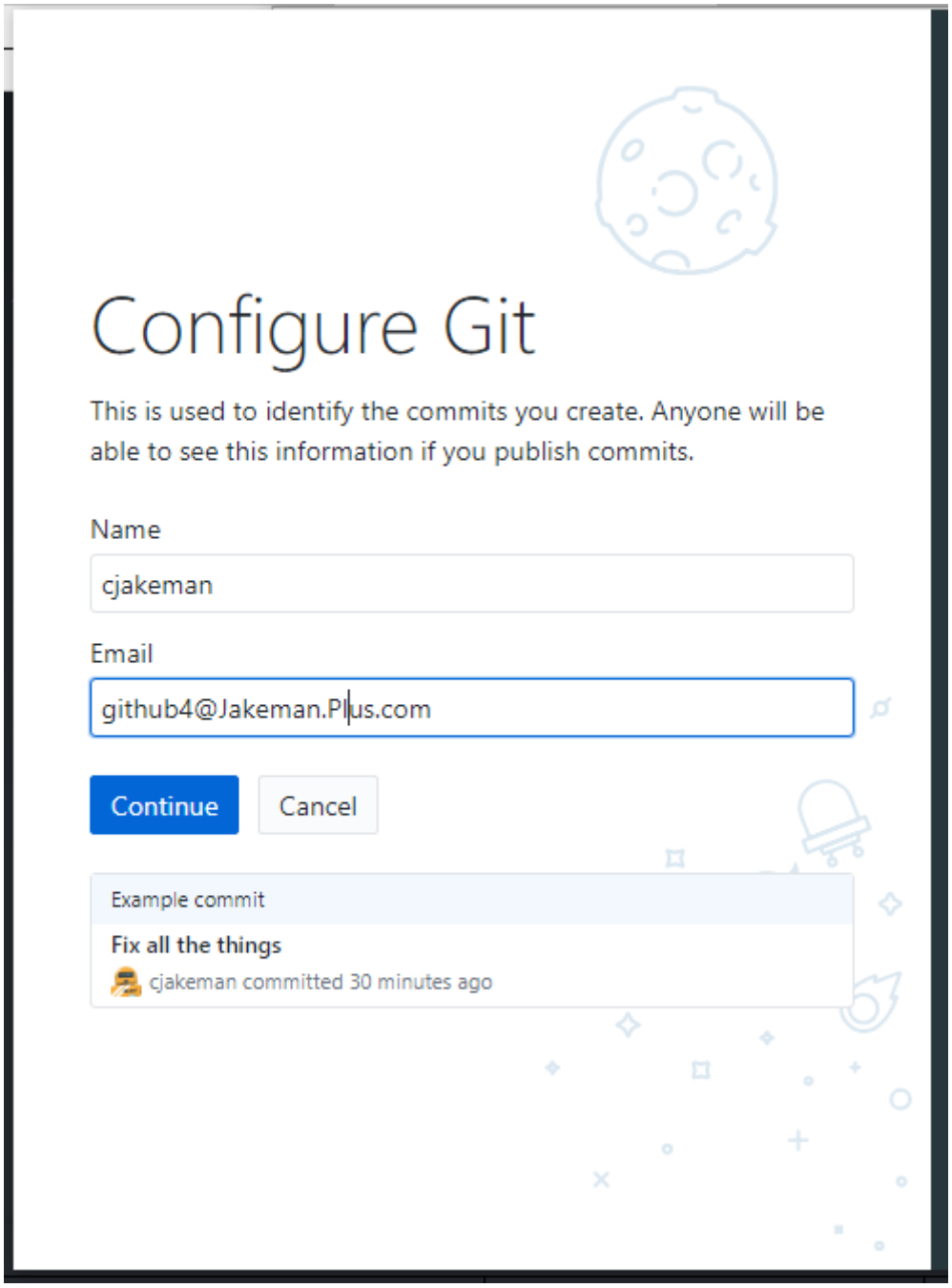
Browse to http://desktop.github.com, click on the *Download* button to download the installer, then run the installer. Choose the free option:



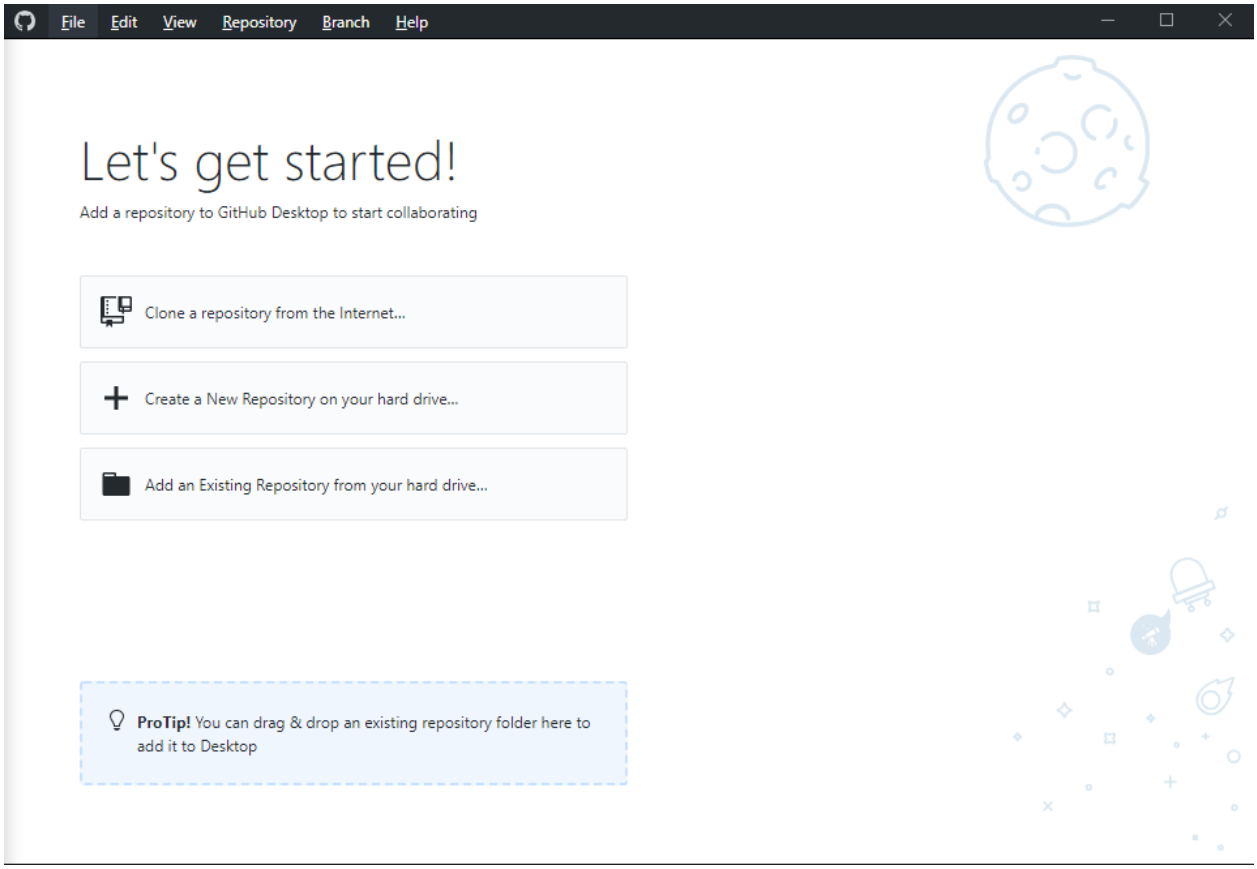
Create an account at GitHub.com



Provide details:



Finally, *GitHub Desktop* opens its launch page:



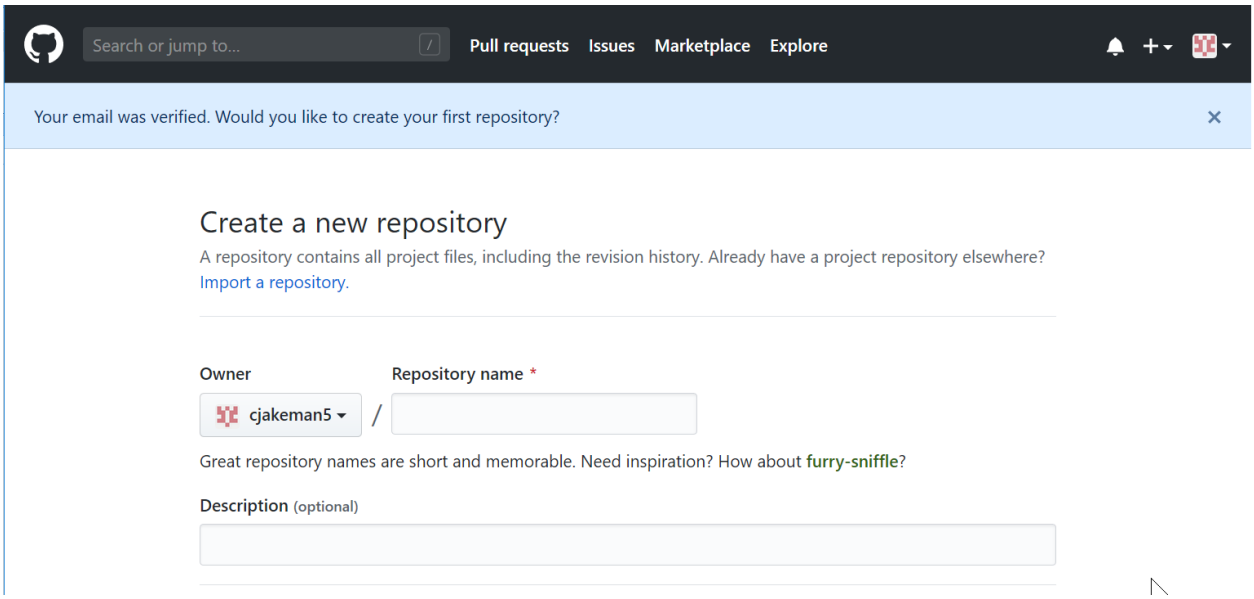
Maintenance Procedure

Initially:

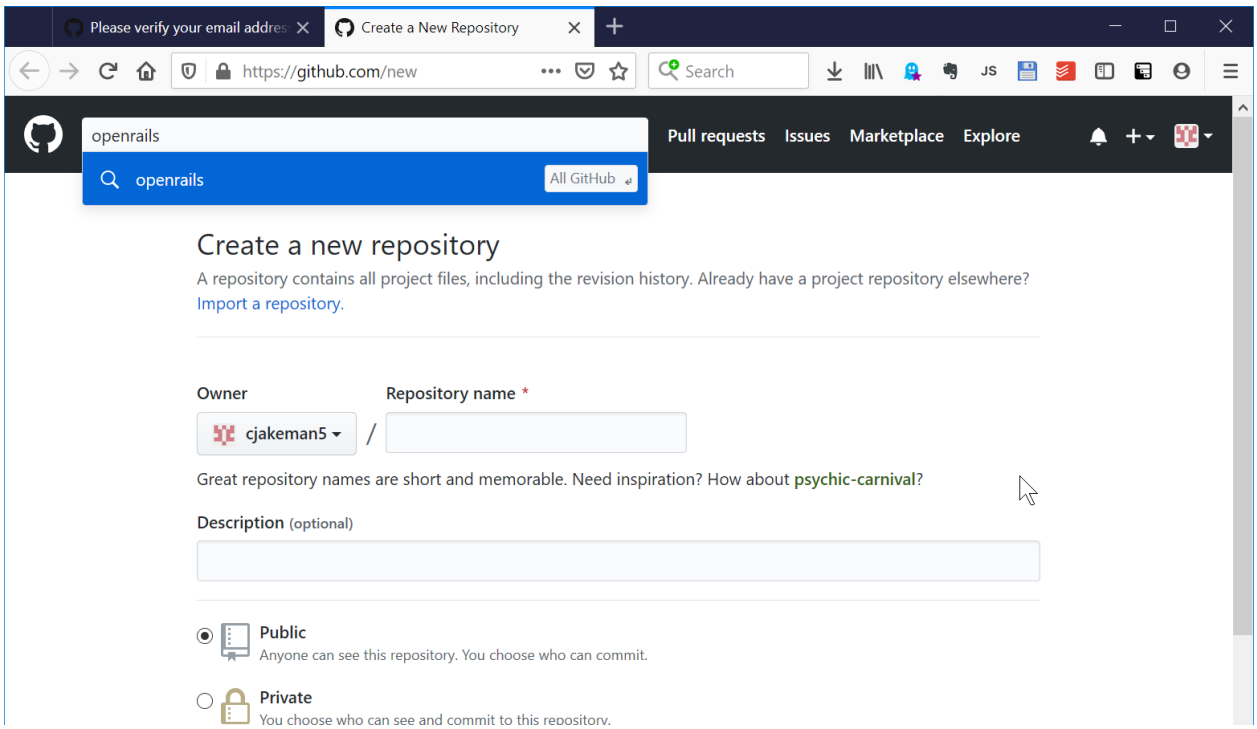
1. Make your own remote repository (or repo) at GitHub

This will be a special copy (called a “fork”) of the official Open Rails remote repo. Git remembers where your repo was forked from and provides a mechanism to check and adopt your changes back into the official repo.

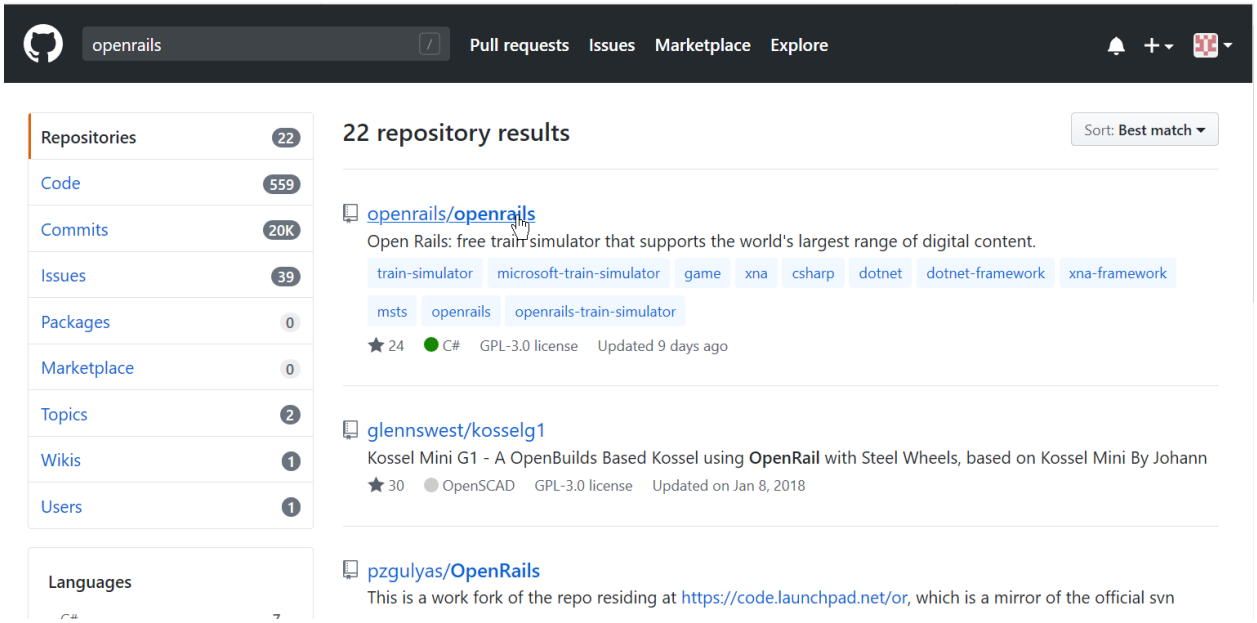
Register at www.GitHub.com and sign in.



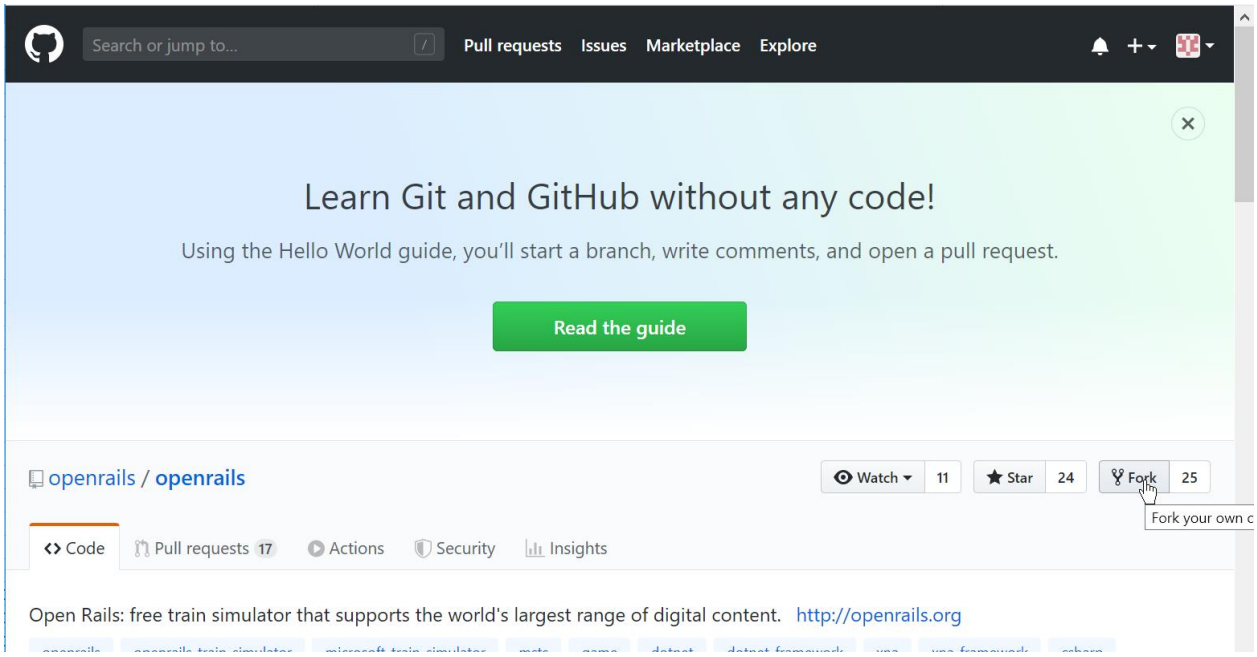
Do not create a new repository. Instead, we want to “fork” the official Open Rails repo, so search for “openrails” or browse directly to https://github.com/openrails/openrails:



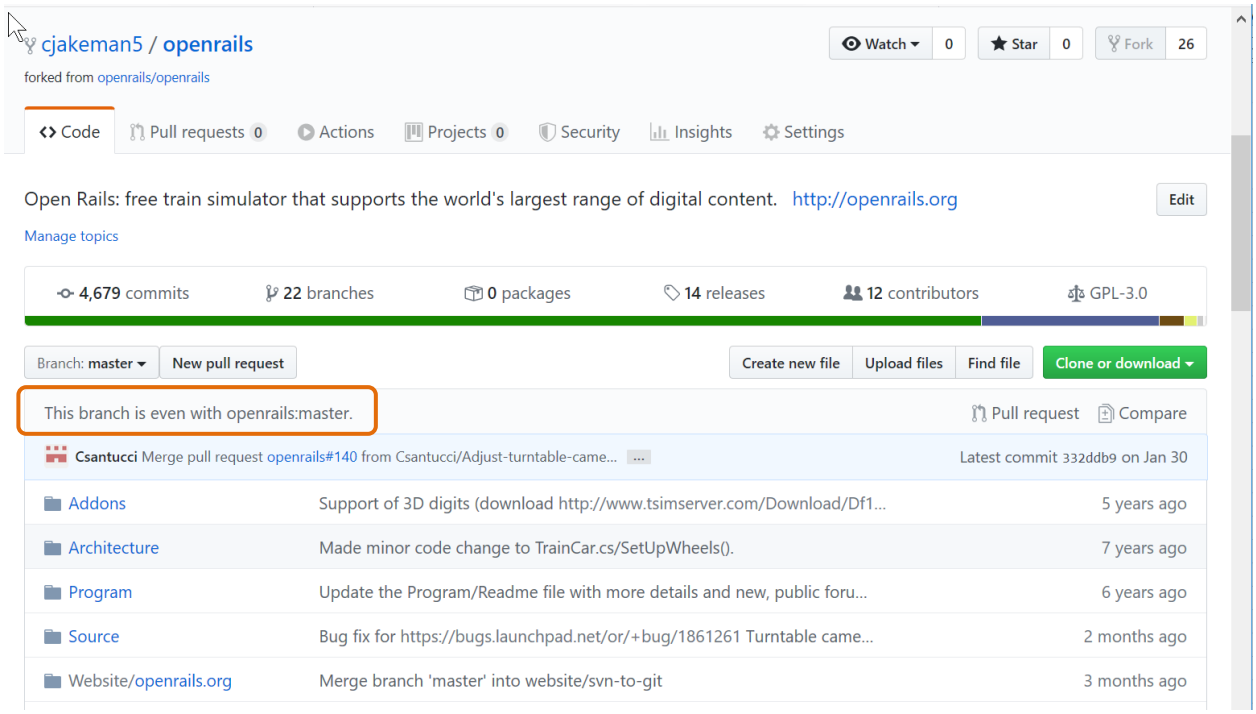
The search will find multiple results for “openrails”, so we click on the official one:



and end up here:



Press the “fork” button and GitHub will copy the official repo to give you a remote repo which remembers from where it was copied.

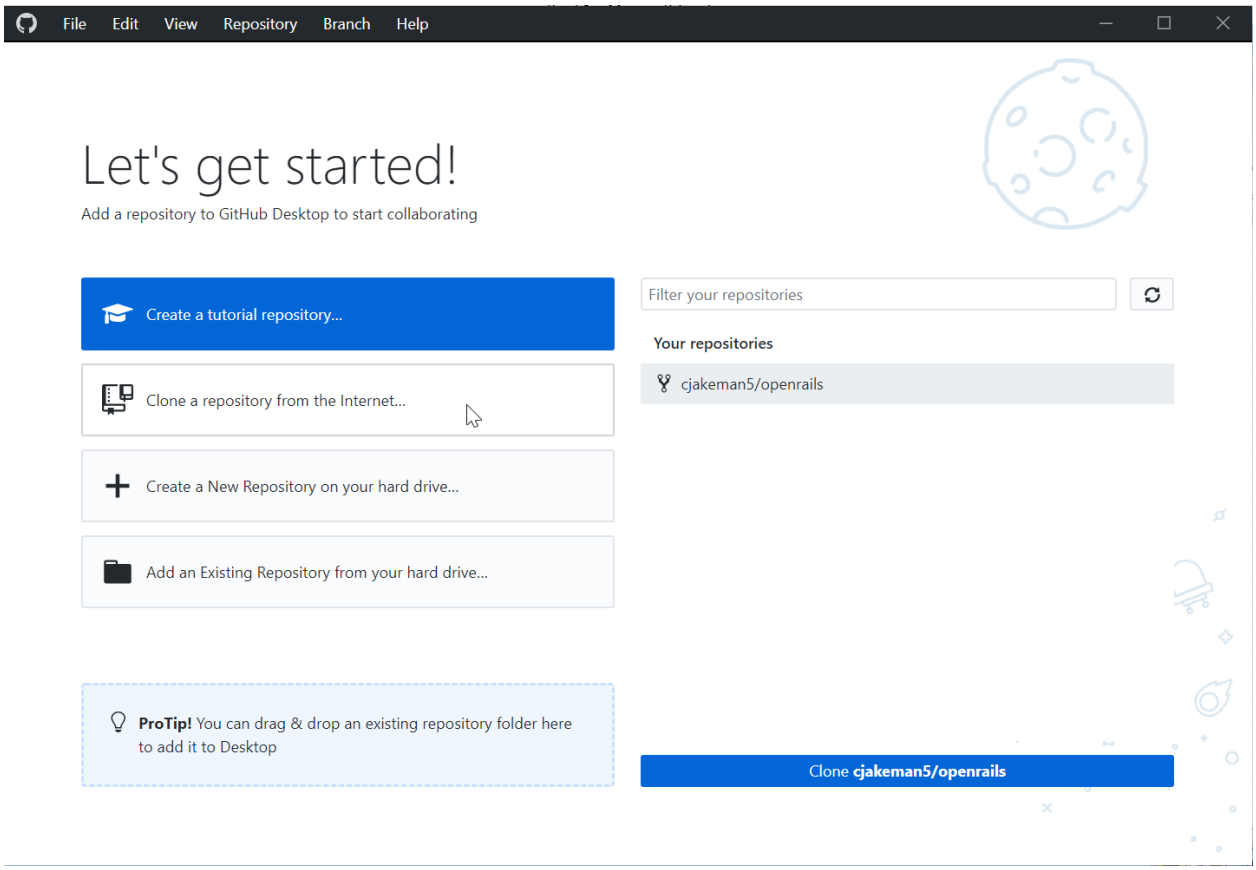


The message “even” shows that your remote repo is up to date.

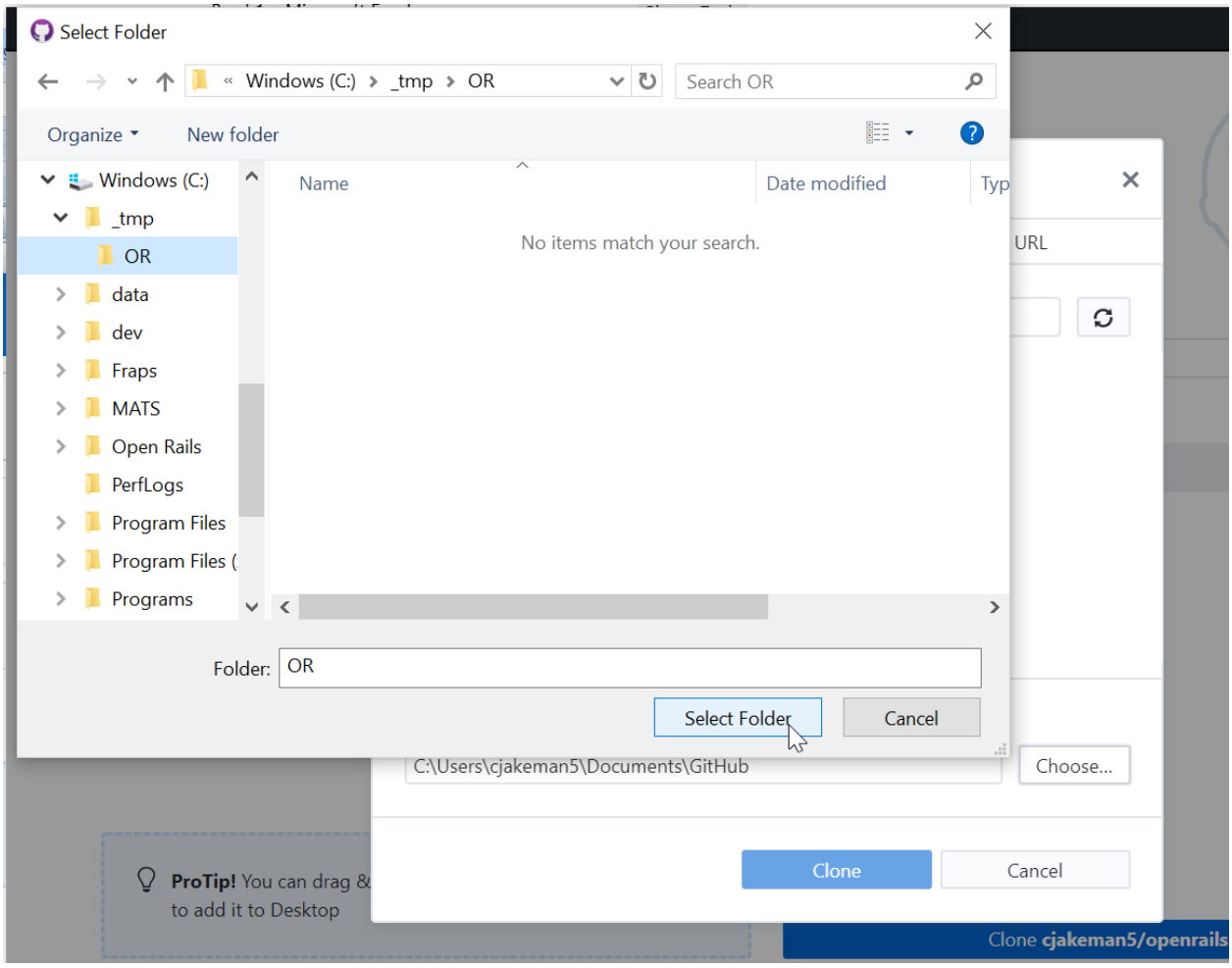
2. Download a copy (called a "clone") of your remote repo

Now return to *GitHub Desktop* where you can "clone" your remote repo from GitHub.com to a local repo on your PC's disk.

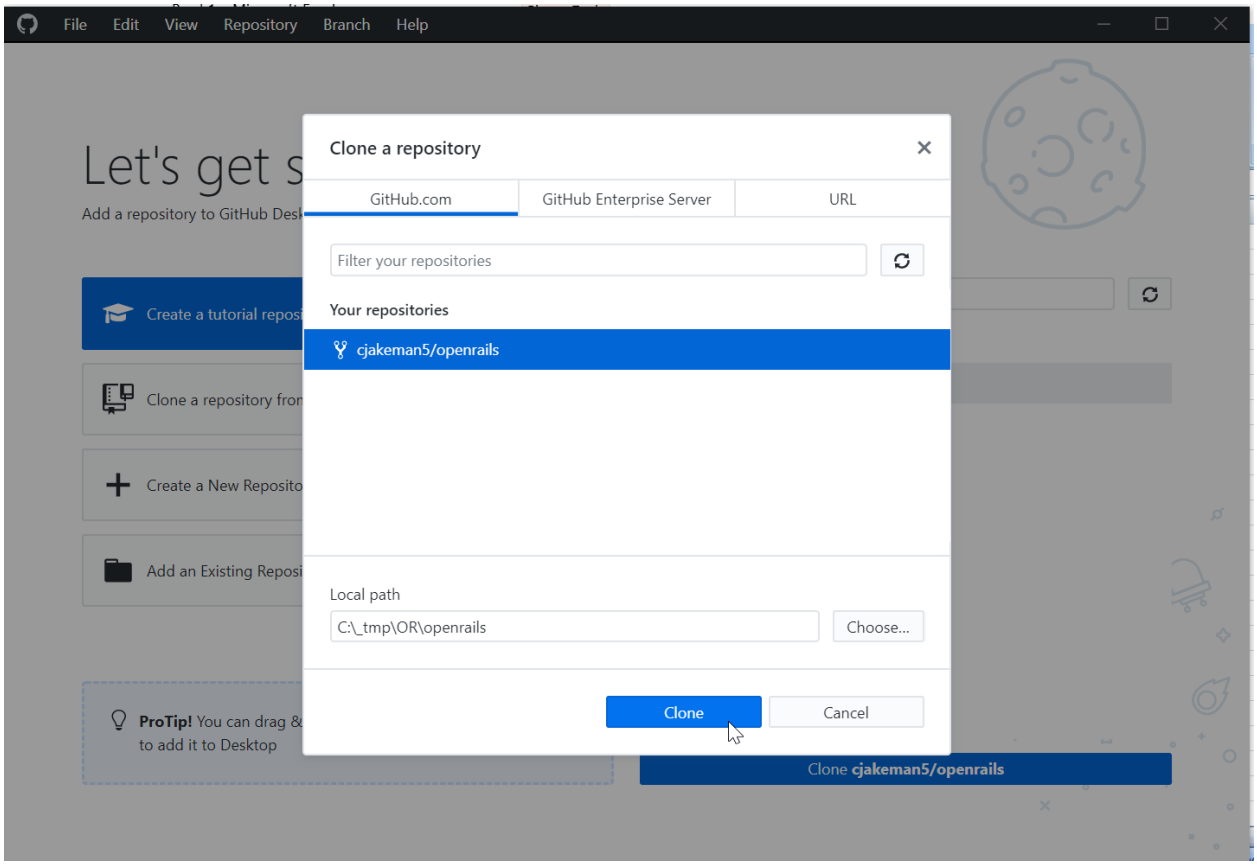
Once again, Git remembers where your local repo was cloned from and makes it easy to push your local changes back to your remote repo.



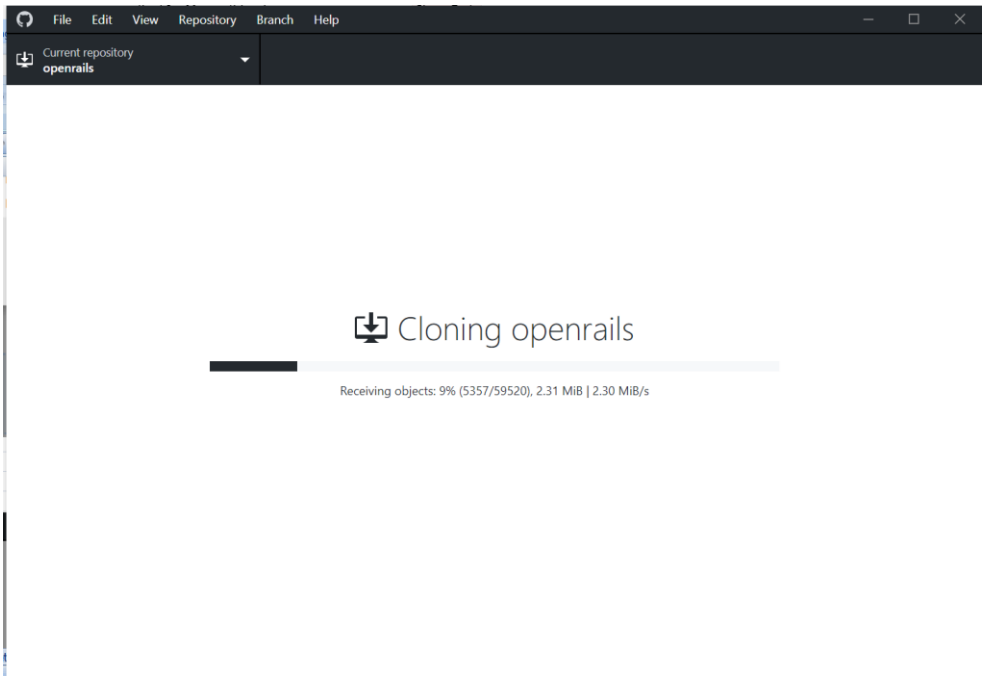
Use *Local path > Choose* to select an empty folder on your PC to hold the local repo:



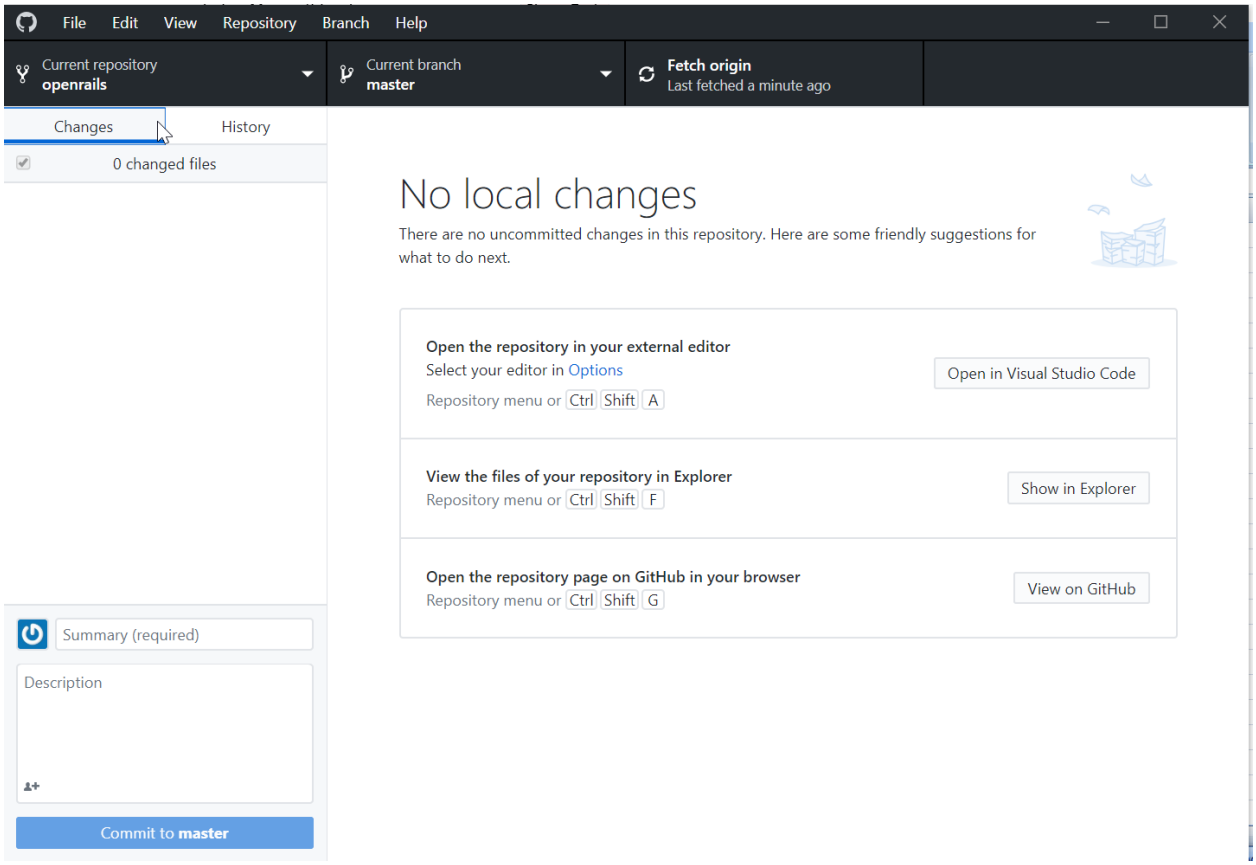
GitHub Desktop will find your remote repo and offer that as “Your repositories”:



Downloading the remote repo may takes some minutes. Subsequent uploads and downloads will be quick as they only transfer the changes.



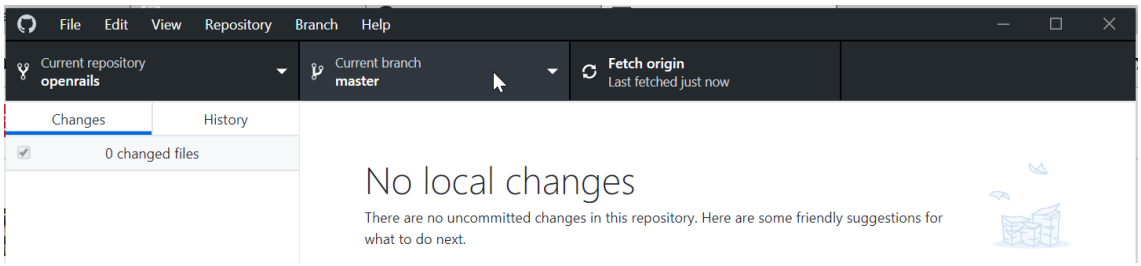
and finally, the system is complete and ready for translation work:



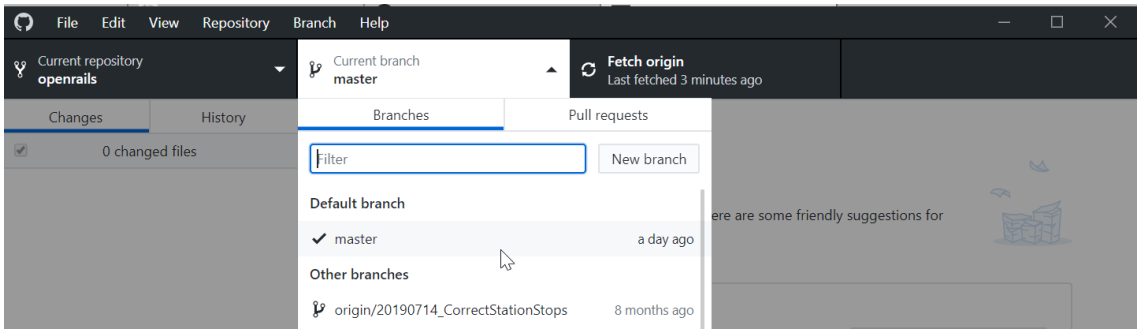
Thereafter:

3. Bring your repos up to date with the official repo at GitHub
Of course, you won't need to do this the very first time.

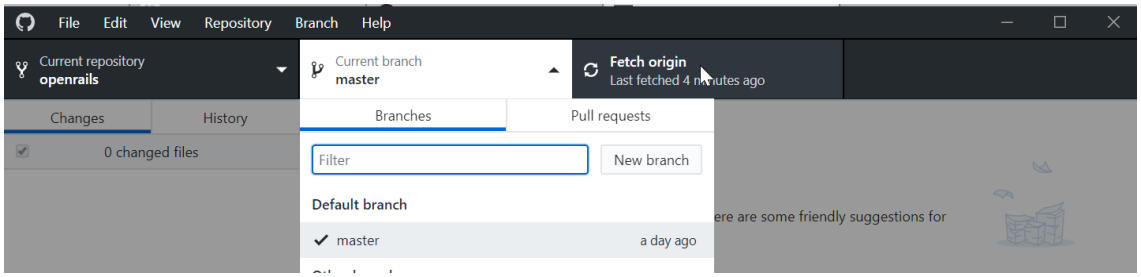
From *GitHub Desktop*, click on the *Current Branch* tab



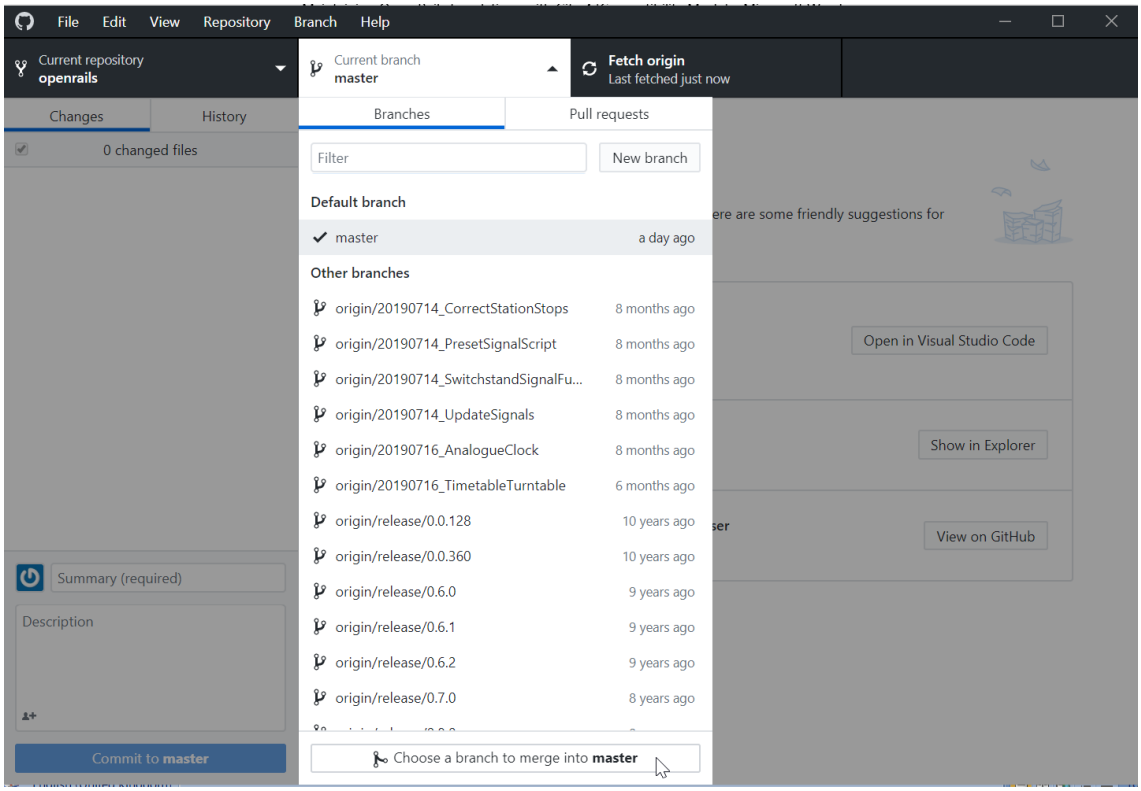
Make sure that the current branch is the "master" branch as we will be updating that from the "master" branch of the official (or "upstream") repo.



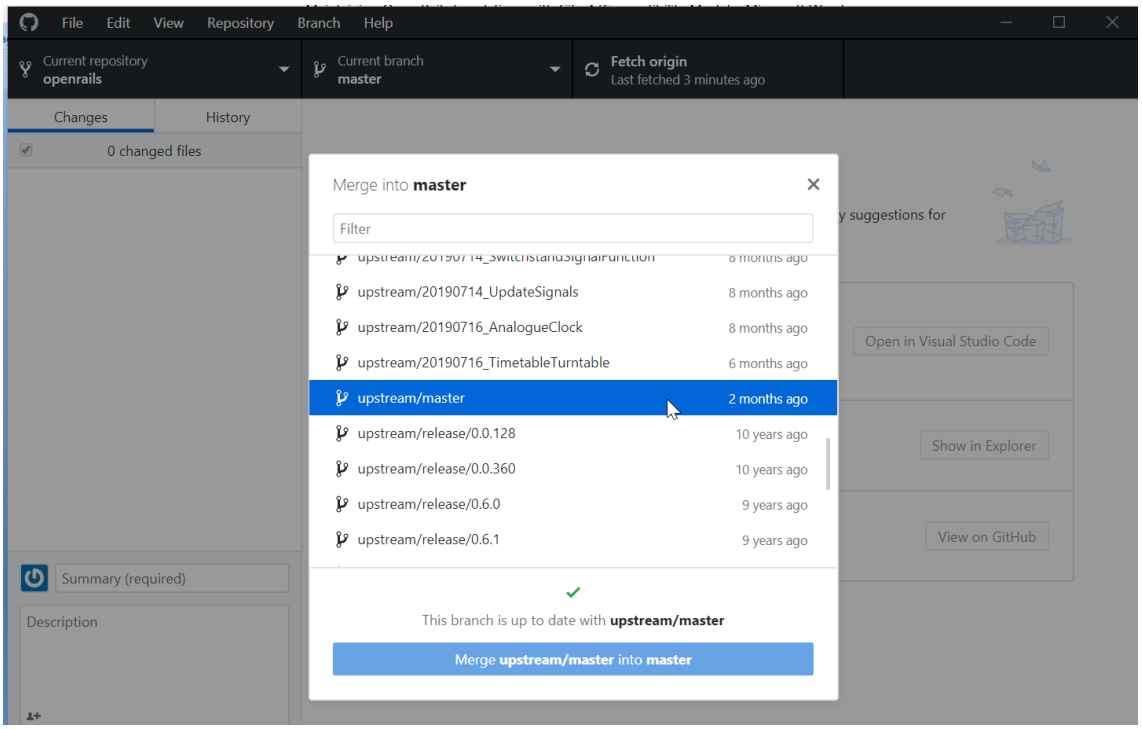
Then click on the *Fetch origin* tab, which downloads any waiting changes from your remote repo ("origin") and also the official repo ("upstream").



Then return to the *Current branch* tab and pick the button at the bottom:



Then scroll down the list of remote branches. This will be the branch from which changes will be merged into your local “master” branch. You should pick the branch called “upstream/master”



If the branch is not up-to-date, then:

1. click the blue button at the bottom to adopt the changes

2. You should also copy these changes to your remote repo so, from the *Changes* tab, press the “Push origin” button to transfer the changes to GitHub.com.

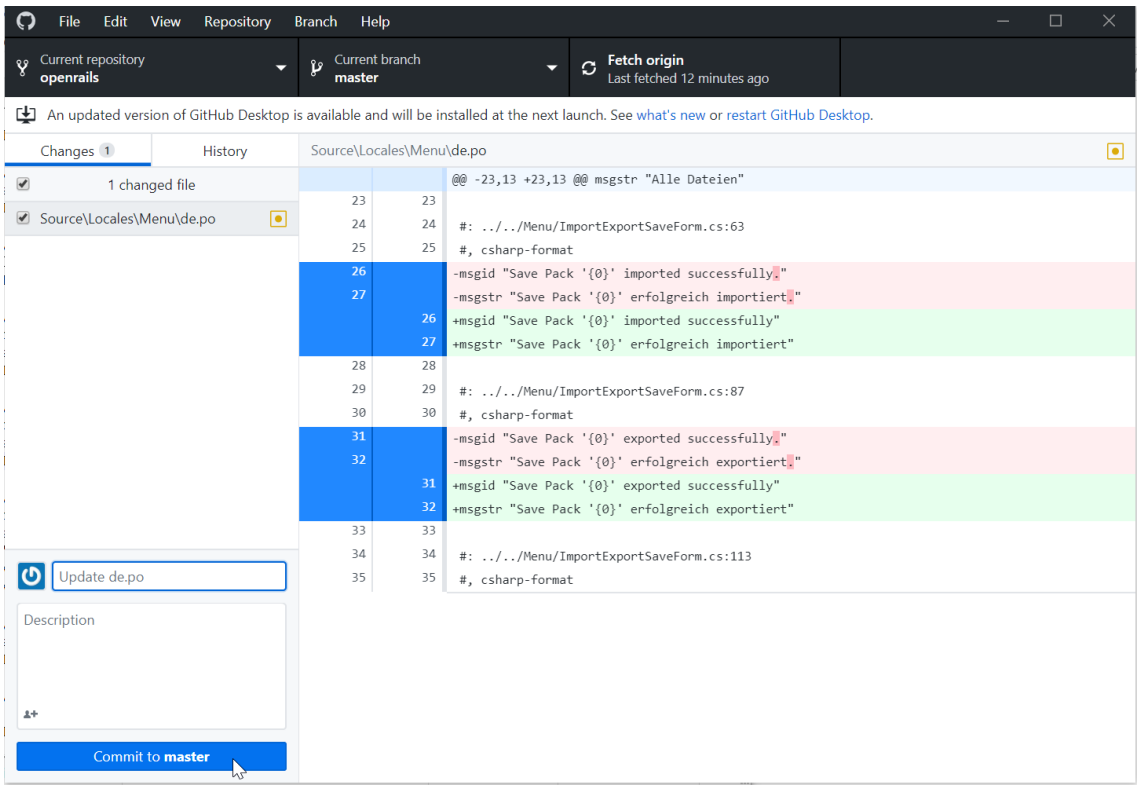
4. Make and test your changes using your translation tools

Follow the instructions in Translating Open Rails[2].

The translation work that you do will change files and possibly add new ones.

GitHub Desktop will track every file in your project that has been added or changed and prompt you to store them locally and securely with a “commit” operation. This gives you a version which you can always go back to.

For example, changes to the German file for Open Rails’ Menu program will be automatically detected once the file is saved. *GitHub Desktop* highlights small changes, provides a default description for the commit and the button *Commit to master* becomes enabled.:



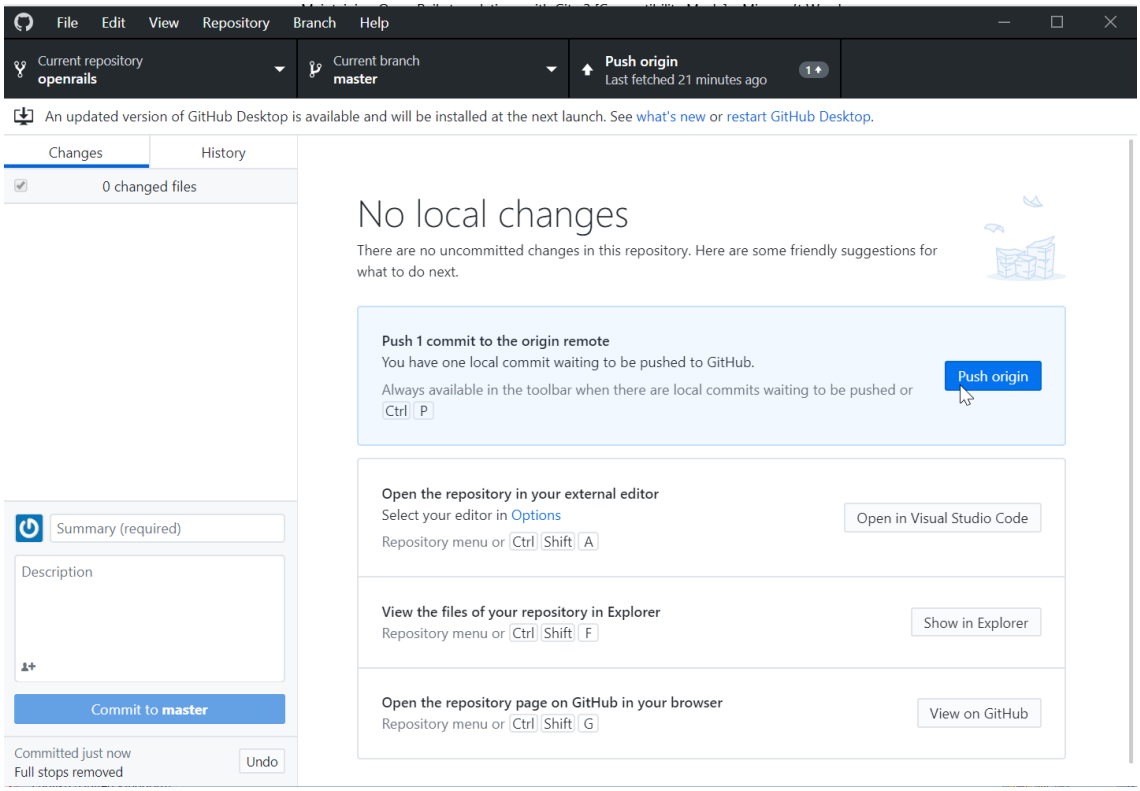
“master” is the branch which is current and to which new commits will be made. You should not need to change this for translation work.

[2] Download from https://static.openrails.org/files/Translating-Open-Rails.pdf

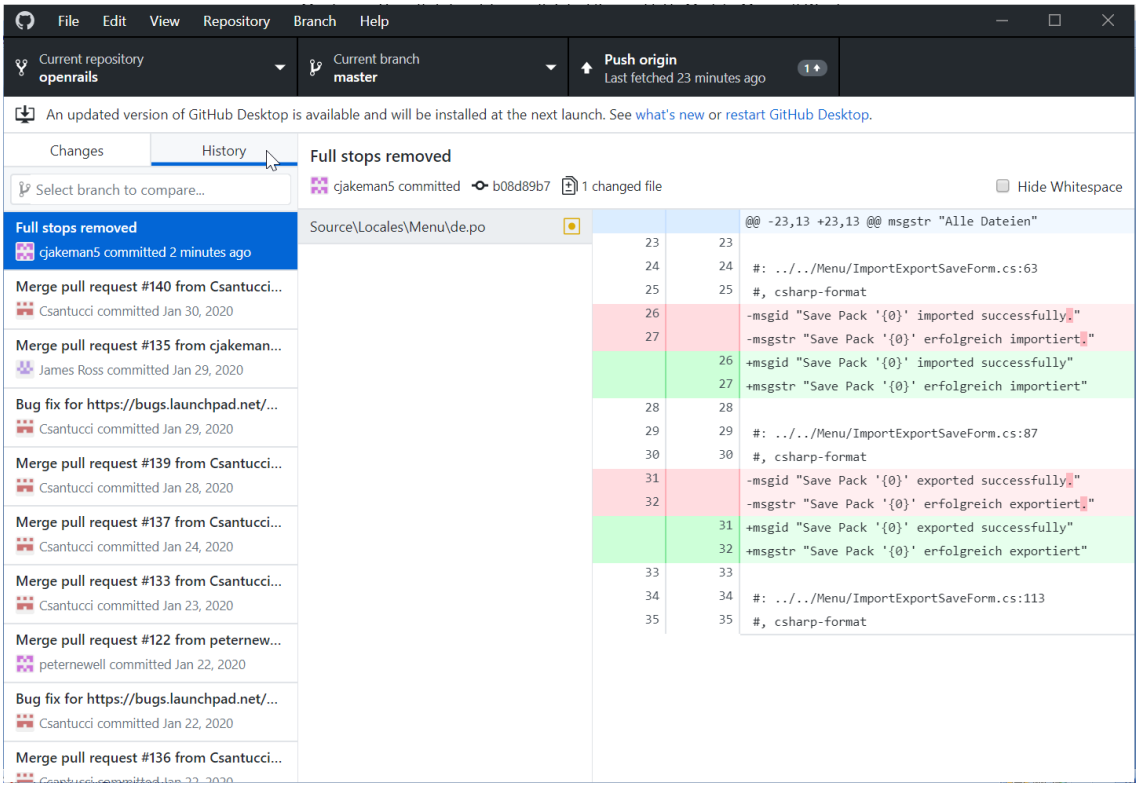
Before committing your changes, change the description from “Update de.po” to something with more meaning.

5. Commit your changes to the local repo on your PC

The “commit” operations stores your changes securely and locally. You can make more changes and more commits if you need to.



By the way, switching from the *Changes* tab to the *History* tab shows all the commits you have made.



6. Push your commit with all your changes
back to your remote repo at GitHub

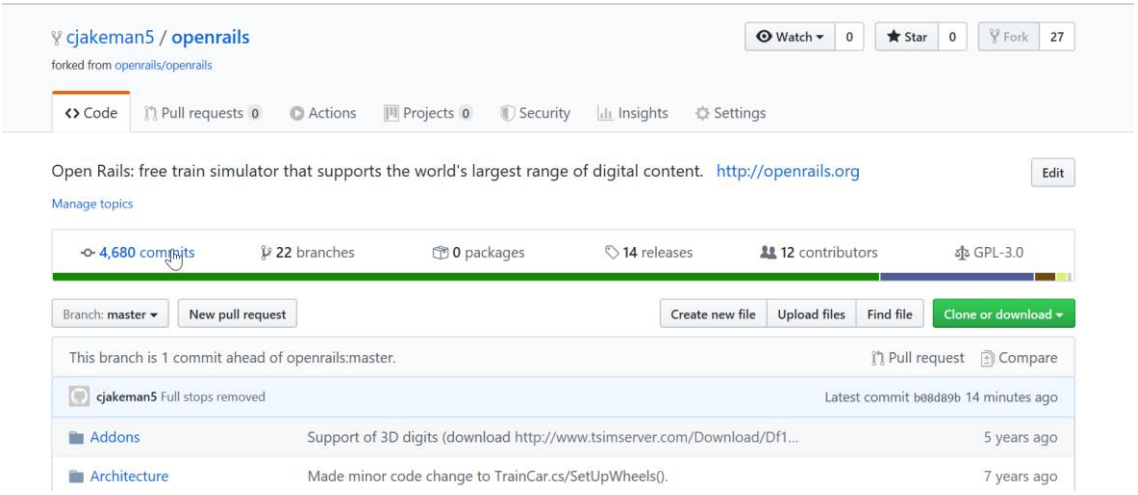
Once all your changes are ready and committed to the local repo, you must “push” them up to your remote repo (which has the default name “origin”).

From the *Changes* tab, press the “Push origin” button to transfer the commits to GitHub.com.

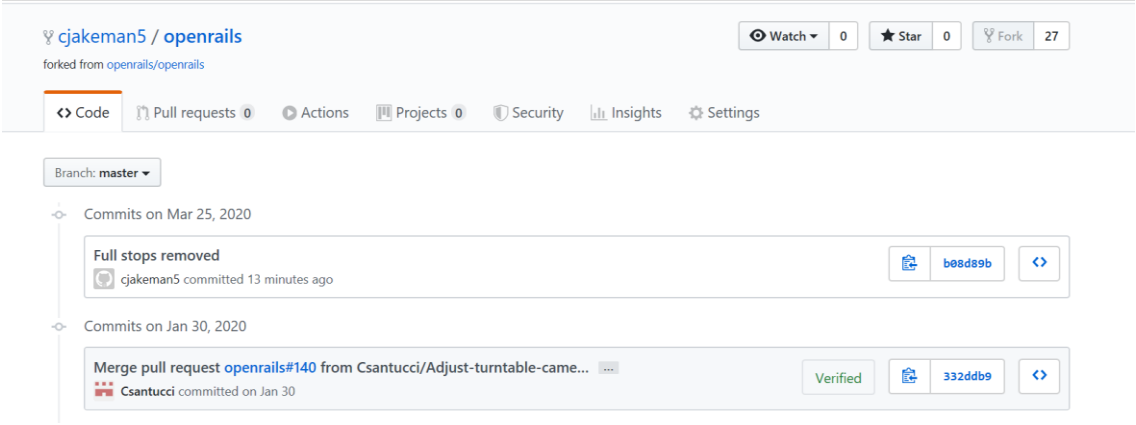
7. Submit your commit by making a “pull request”
to the official repo at GitHub

Return to your remote repo at GitHub.com.

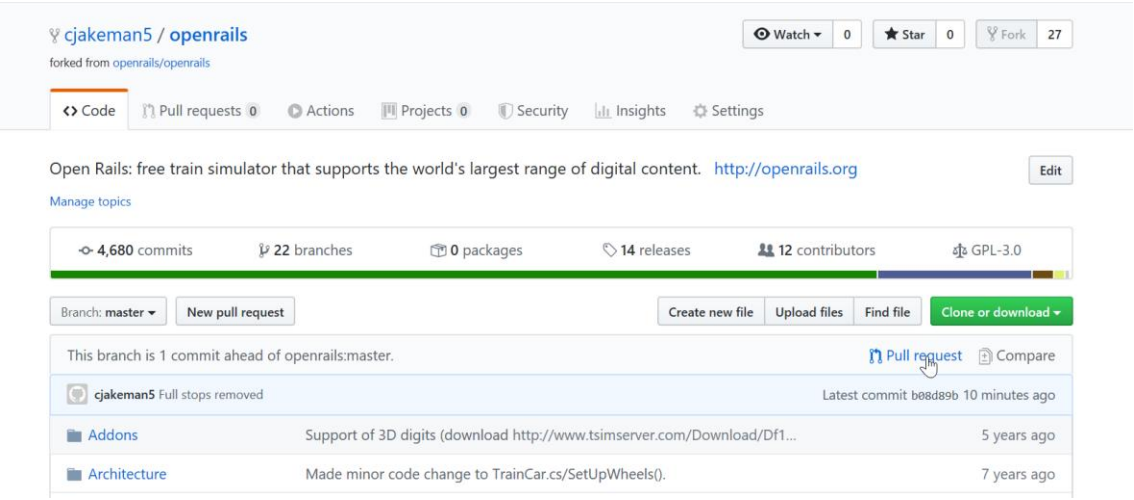
You will see that your remote repo is 1 commit ahead of openrails:master (the “master” branch at the official repo).



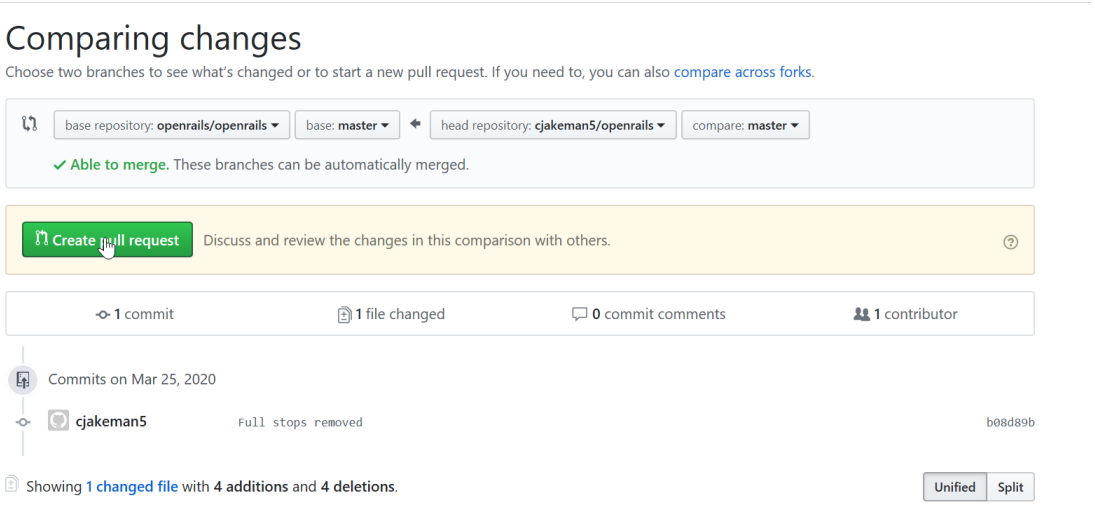
If you press the blue commits link, then you get to see that your new commit has arrived:



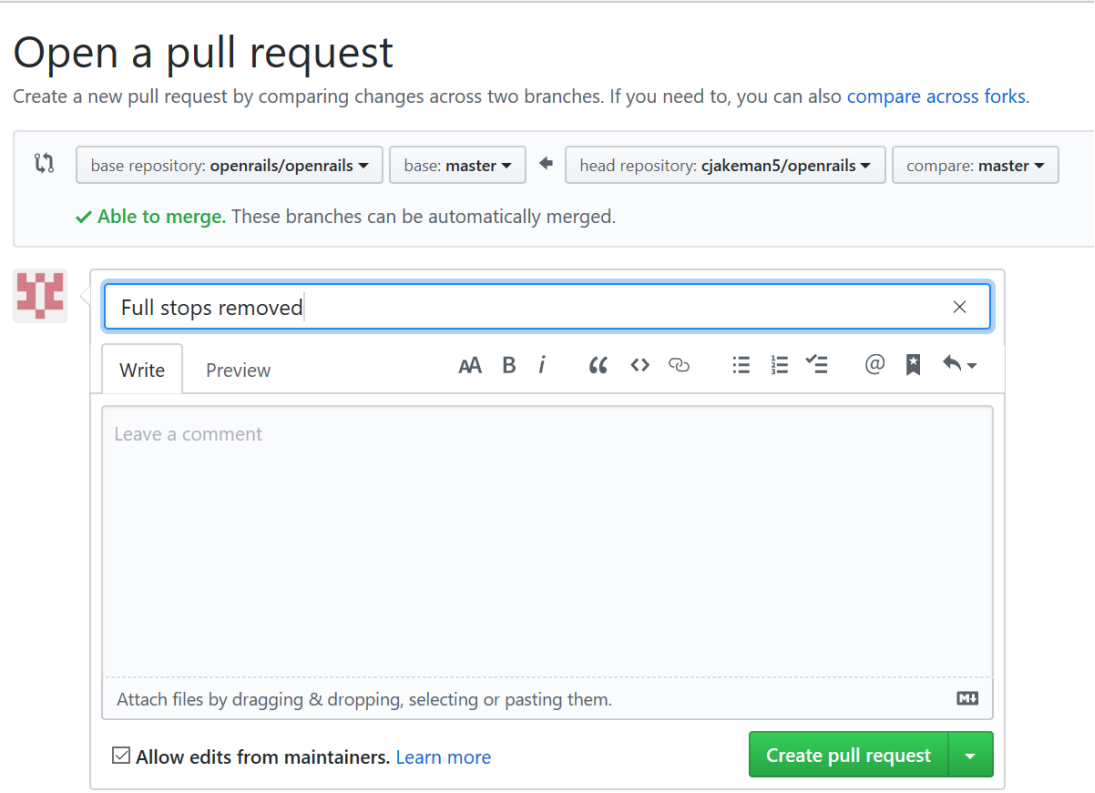
Then go back to the previous page and press “Pull request” to submit your change to the official repo.



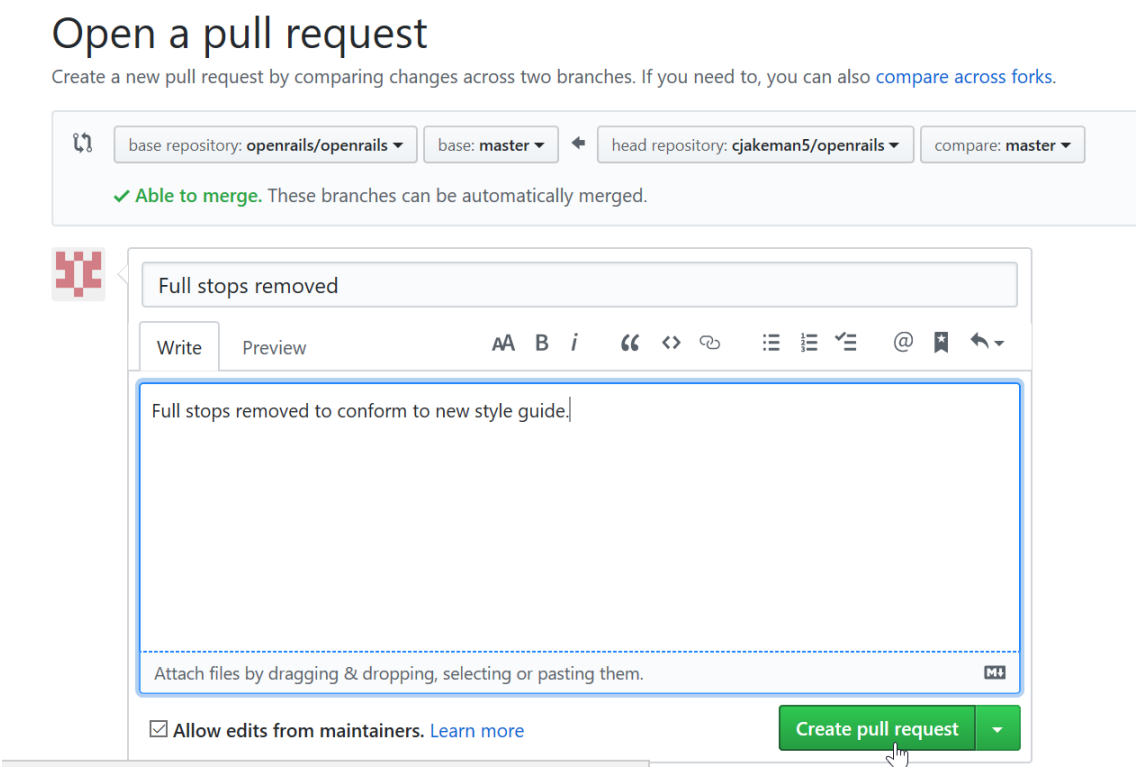
GitHub will show you your files and check that a merge with the official repo is possible without conflicts. When translating, there is unlikely to be a conflict unless two translators are working on the same language at the same time. In that case, it would be better to share the same remote repo.



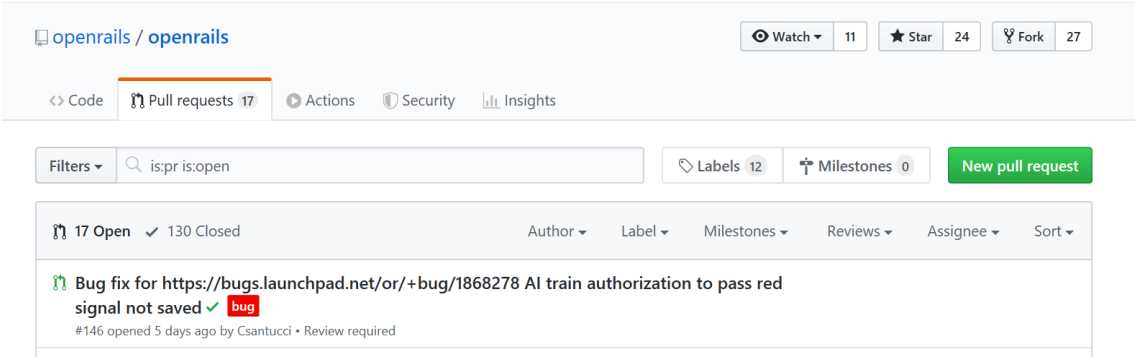
If you want to provide an explanation of your work to help the reviewer, then use the form to “Leave a comment”.



Finally, press the green button:



Your new Pull Request will be added to the top of the list at the official repo:



Job done !