Users Guide
Publish software easily in GitHub

Miguel Cruces

LabCAF / IGFAE
Universidade de Santiago de Compostela

1 GitHub briefly

GitHub is a provider of Internet hosting for software development and version control using Git. It offers the distributed version control and *source code management* (SCM) functionality of Git, plus its own features. It provides access control and several collaboration features such as bug tracking, feature requests, task management, continuous integration and wikis for every project. Ironically, it is not a project based on free software, it is a subsidiary of Microsoft.

It has a webage very well *self-documented*, with detailed information about the operation and a complete help guide.

2 How to publish a repo?

There are several ways to create a repository on GitHub, but here we will explain the two most common: starting from scratch, and starting with a previously initialized local Git repository.

2.1 From scratch

The easiest way to create a repository is by starting from the website. Once you have your account[1], you can follow the next steps in the GitHub webpage.

In the upper-right corner of any page, click on the plus + sign to display the drop-down menu and then select New Repository; then a section will appear like the screenshot shown in Figure 1.

[1]Please, if you don't have a GitHub account yet, join here.

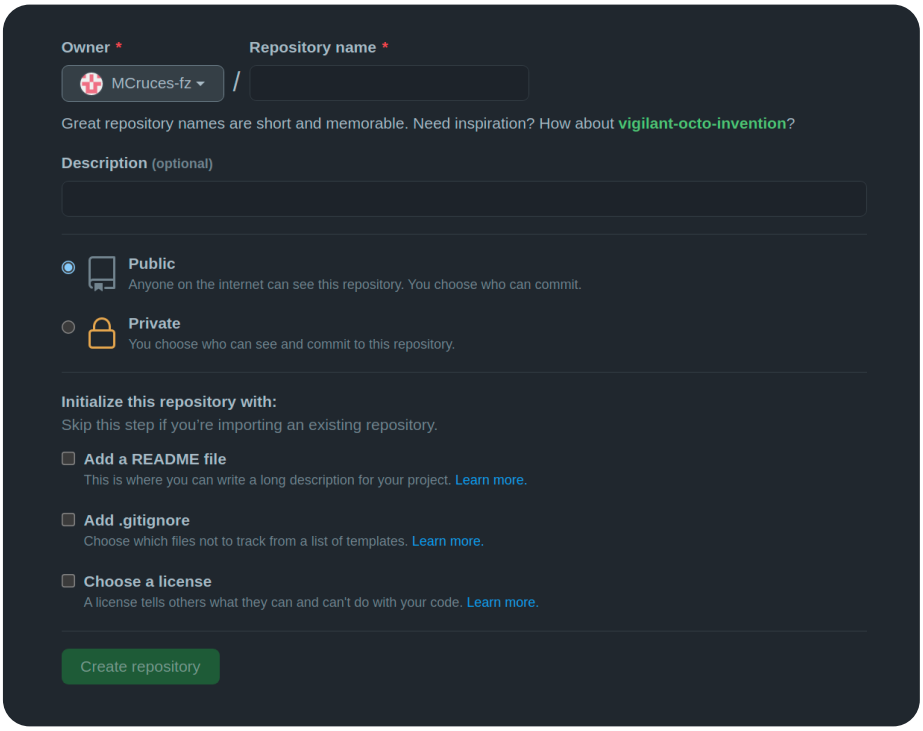

Figure 1: Example of creating a repository on GitHub.

Below the Owner drop-down menu you can choose your own user, or a group to which you belong. As we all know, it is difficult for scientists to choose original names and that is why *Octocat* always tries to inspire us with a "short and memorable" name for our repositories under the Repository name entry.

Writing a Description is always recommended, although optional. This will help the user understand the subject of the project before clicking to read the code.

As you can see, it is only mandatory to fill in these two fields: Owner* and Repository name*, but in terms of visibility there are only two options and one of them must be chosen. If we are trying to create a open-source repository, I think it goes without saying that we should choose the Public option.

GitHub helps us by giving us the option to automatically create a series of plain text documents:

- *README.md*: A markdown file which will be displayed in your repository. Here you can write a more detailed description for your project.
- *.gitignore*: This is a *dotfile*[2] where you can write which files not to track. Once you click on the checkbox, a drop-down menu will appear and you can choose a *.gitignore* template[3].

[2]Those are hidden files in Linux of the form "*.filename*", for instance all your configuration files in your home directory: `~/.*`.

[3]Of course, each type of project has a series of files that do not need to be shared, since they are generally compiled, executed...

- *LICENSE*: Here you can choose a license for your repository, but we will see it in more detail in the next section.

Before clicking Create repository, you can change the default branch name to `master` instead of `main` here.

2.2 From a local Git repo

Maybe you have a local project and you want to publish it. If so, you can do it cleanly by following these steps.

First follow the steps written in previous section "From scratch", but stop once you had choosen the visibility of the repo. Don't add any files! as you can see, *Octocat* suggests you skip that step if you're importing an existing repository, because when you push your files there will have merge conflicts — and believe me you don't want that. Once this is done, you can Create repository and you will see some help commands.

Open your favorite terminal emulator and `cd` into the root directory of your project. Then, initialize a git local repository as follows

```
cd /path/to/root/dir/
git init # Initialize git repo (a directory .git/ appears)
git remote add origin https://github.com/<username>/<reponame>.git # Add
    remote
vim .gitignore # Edit a .gitignore if you want
git add --all # Track all files except .gitignored
git commit -m "Mi first commit" # Write a commit message
git push -u origin master # Push your new branch to GitHub
```

Finally, if you refresh the webpage you will see your ***pushed*** repository with a nice readme below —or an invitation to create it, failing that.

But keep this in mind! If you add a readme from the GitHub webpage, a new commit will appear in the history of your `origin`, hence you must pull the changes to rour local repository:

```
git fetch origin master # Check updates
git pull origin master # Download and merge updates
```

Learn more about these commands:

```
man git fetch
man git pull
man <any_command>
```

2.3 Magit

As is written in the *CONTRIBUTING.md* of next-exp/IC project:

" Do yourself a favour, and use magit to interact with git: it's the best git UI on the planet. Even if you don't use or dislike Emacs, use magit, and think of Emacs as the GUI framework in which magit is written.

You will enjoy and learn to understand git much more if you use magit.

[Editor's note: Seriously, I tried to find something to recommend for those who don't use Emacs. It wasted a lot of my time, and I came to the conclusion that recommending anything else would be a waste of your time. Just use magit.]

To make the magit experience even better, use helm. "

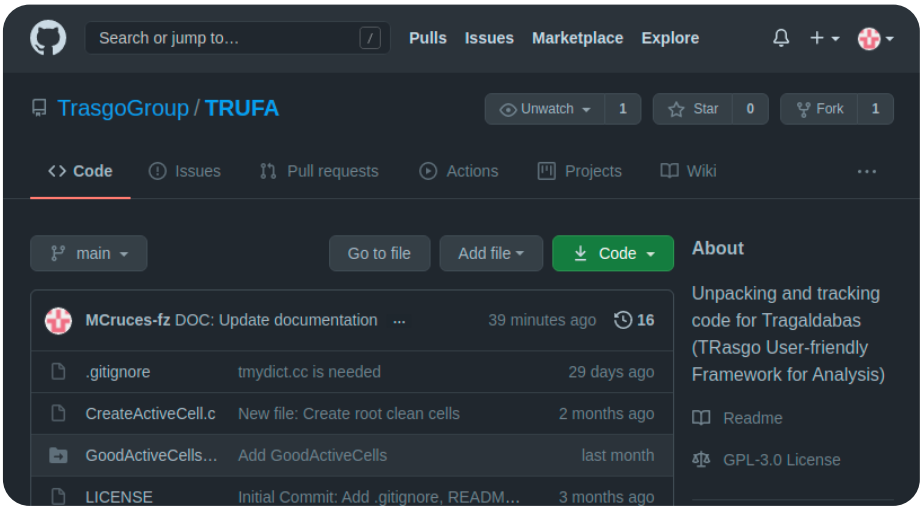

Figure 2: Example of main page of a repository.

This tool is very useful to interact with git. The easiest GUI maybe is *GitHub Desktop*, but it isn't powerful enough.

3 How to write a License

You can include an open source license in your repository to make it easier for other people to contribute.

On GitHub, navigate to the main page of the repository to which you want to add the license. Check Figure 2, and to the left of the green button Code, click on Add file > Create new file. In the file name field type "*LICENSE*"[4] and to the right of the file name field a new button will appear: Choose a license template. Click on it!

On the left side of the page, under Add a license to your project, review the available licenses, then select a license from the list, click Review and submit and finally commit changes.

Again, remember `git fetch` and `git pull` into your local repository.

Once you have your *LICENSE* file commited, you can add a small piece of text that refers to the license in the header of each file to which said license is applied. If your license allows other developers to edit your code, they must edit the header of the files that are modified following your guidelines.

For more help, visit Adding a license to a repository

4 Fork: Contribute to an existing repo

Navigate to the repository which you want to fork using your web browser, and fork it to you account by clicking in the Fork button, shown in Figure 2. The button is in the top-right corner of the page. If you belong to any organization,

[4]With all caps recommended, but with small letters it will work.

GitHub will ask you *Where should we fork "reponame"?* and you must choose the profile where you want to have said repository.

Once you have forked it, you can navigate to your GitHub profile and go to your repositories. If you open the forked repository you will see something similar to Figure 2, then you are able to clone your forked repository by clicking in the green button Code and copying the HTTPS link.

You can clone this reposiroty from your GitHub account:

```
cd <work_dir>
git clone <copied_link>
```

directly. Now you can edit, `git add`, `git commit`, `git push` and all you want.