



# 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 webpage 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<sup>1</sup>, you can follow the following 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.

---

<sup>1</sup>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.

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 an 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*<sup>2</sup> 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<sup>3</sup>.

<sup>2</sup>Those are hidden files in Linux.

<sup>3</sup>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 chosen 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 be merge conflicts (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

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

## 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. ”*

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 contribute to a existing repo

blah blah

## 4 How to write a License

Blah blah