

GitHub Exercices
26/06/2025
Rosalie Bruel (OFB, pôle R&D ECLA)

Tutorial modified from a previous intro to GitHub (Rosalie Bruel & Jerome Mathieu, GitHub Exercices, Club iEES-Paris, 1/12/2021)

Resources for an in depth understanding of Git (not updated since 2021):

<https://gist.github.com/peterhurford/4d43aa5d6de114c0c741ba664c9c5ff5>

<https://docs.github.com/en>

glossary : <https://docs.github.com/en/get-started/quickstart/github-glossary>

Intro : what is GitHub.....	2
Set up your GitLab account.....	4
I - Get started with the GitHub online interface (online repository).....	4
1.1° Create an account on GitHub.....	4
1.2° Get familiar with your space.....	4
1.3° Find a public repository.....	4
1.4° Explore a GitHub repository.....	4
1.5 Explore the history of changes in a repo.....	5
II - Work with your own repositories.....	8
2.1° Create your own repository.....	8
2.2° Do a local modification in the repo: "commit".....	8
2.3° send your modification on GitHub: "push".....	8
2.1° Create your own repository.....	10
Option 1 : Create your own git folder in your computer and publish it to GitHub.....	10
Option 2 : create on Github.com then add it to your computer.....	11
2.2° Do a local modification in the repository: "commit".....	13
2.3° Send your modifications on GitHub.com : "push".....	13
2.4° Check if any local repository has a commit waiting to be pushed.....	13
In GitHub Desktop, click on the triangle on the side of the current repository (1).....	14
2.5° Compare you local folder to the online version and update you local folder.....	14
2.6 Remove / delete a folder.....	14
III - Work with others.....	15
3.1° Visualize multiple branches - what your project may look like with multiple collaborators.....	15
3.2° Issue, fork and pull request.....	15
3.3° Make a pull to a public repository.....	15
3.4° Contribute to repository as a contributor.....	16

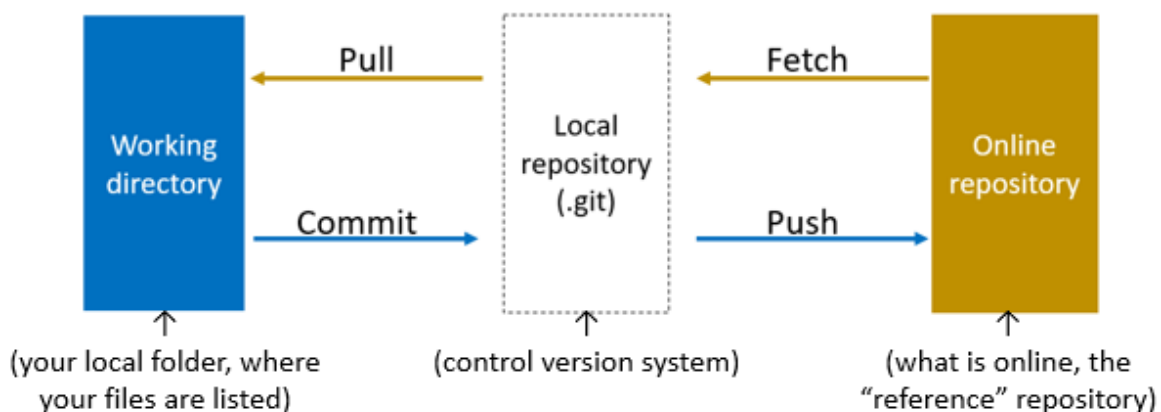
Intro : what is GitHub

Roughly speaking, GitHub can be seen as online folder, just like a dropbox or a GoogleDrive. You can upload online documents in folders (called “**repositories**”) and download them. If the repository is open or shared, other people can also download, upload the documents and make modifications in the files.

But there are also strong differences with a traditional online folder. To start, GitHub is dedicated to share code. Consequently there is a **versioning system** to track the history of the folders and of each file, which make sure that the changes in the files are controlled and trackable (you have access to all previous versions of a file and of directories).

So there is a kind of back and forward movement of files from local computers to the online folder, like in other online folder systems, but there is in addition the control version system software that operates in your computer as a kind of buffer between your computer and the online folder. (You can see the .git folder if you display your hidden folders)

GitHub Desktop terminology



Due to this extra layer in the downloading/uploading system, different steps take place:

- Making yourself a modification of a folder or document in your computer and making it official (kind of new version) in your control version system is called a “**commit**”.
- Uploading a file from your controlled system to the online folder is called “**pushing**”.
- Downloading to its own computer the last modifications of the online files is called “**fetching**”.
- Accepting modifications from the online repository and propagating them to your files on your own computer is called “**pulling**”.

This system implies several things :

- The authoritative files are the last ones that where approved by the admin. You can think of them as the files in the online folder.
- If you want to make a change to the files, you need to have them in your computer, make change in the files, validate locally the modifications (“commit”) in your local computer, and then upload (push) the modifications to the online folder. So compared to traditional dropbox system, there is one additional step between the local modifs and the upload, which is the commit step. (See details in section II).
- Only the admin(s) can accept the modifications straightaway. Other people can only propose modifications based on their own copy of the files (called “**branch**”). When

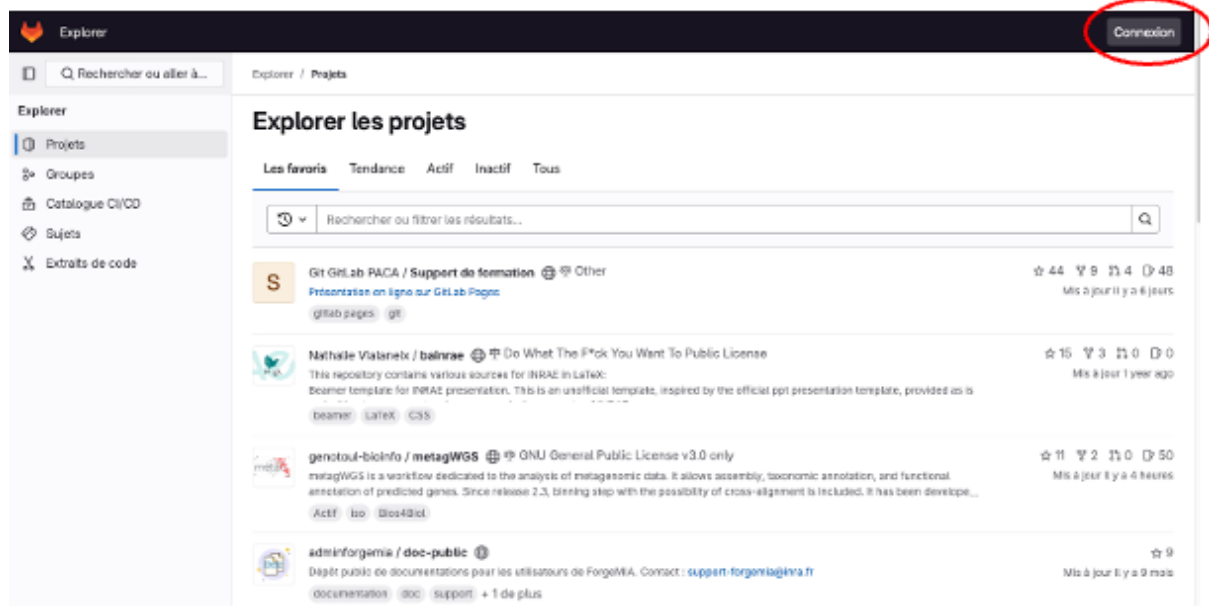
they want to propose an update, they need to send a request, which is called a **pull request** (section III) to merge their branch into the main branch.

- Lastly, GitHub repositories can be directly published on Zenodo and get a DOI, which makes the code citable.

Set up your GitLab account

The institutional forge of INRAE has been in production since June 10, 2025 - it .

<https://forge.inrae.fr/explore/projects/starred>



I - Get started with the GitHub online interface (online repository)

1.1° Create an account on GitHub

<https://github.com/>

1.2° Get familiar with your space

In the top right of the web explorer,

- go to “your profile” → this is your public profile
- go to “Repositories” → This is where all your repositories will be accessible

1.3° Find a public repository

- Click on the github logo in the top left of the screen
- Do a search in the search bar.
- Clic again on the GitHub logo and do a search for “LakeEnsemblIR”

1.4° Explore a GitHub repository

- Go to the LakeEnsemblIR repository (Hint : see question 3 to find it)

The screenshot shows the GitHub repository page for 'LakeEnsemblR' by 'aemon-j'. The page is annotated with red circles and numbers 1 through 7:

- 1**: Points to the file list on the left, including folders like .github, R, data-raw, data, inst, man, pkgdown/favicon, tests, vignettes, and files like .Rbuildignore, .gitignore, DESCRIPTION, LICENSE, NAMESPACE, NEWS.md, README.Rmd, and README.md.
- 2**: Points to the commit message column, such as 'Add write permission', 'Update linking to website rather than wiki', 'Added met_var_dic to internal data so it ...', 'added possibility to output heat fluxes', 'Add gitignore', 'Update linking to website rather than wiki', 'Add pkgdown favicon', 'Add skips for Mac on FLake & Simstrat', 'Fix cross ref', 'Add articles to ignore', 'Add folder which is used for examples', 'Switch URL to aemon-j', 'Update LICENSE', 'Merge remote-tracking branch \'upstrea...', 'Add NEWS', and 'Update linking to website rather than wiki'.
- 3**: Points to the commit date column, showing dates like '3 months ago', '2 years ago', '3 years ago', and '5 years ago'.
- 4**: Points to the merge pull request link: 'Merge pull request #291 from...'. Below it, the commit hash 'bd60674' and the time '3 months ago' are visible.
- 5**: Points to the commit count '1,400 Commits'.
- 6**: Points to the 'About' section on the right, which describes the package as 'An R package that facilitates multi-model ensembles for lake thermodynamics. Also includes tools for calibration, sensitivity analysis and data visualization.' It also lists the repository URL, license (GPL-2.0), and other details.
- 7**: Points to the 'Contributors' section at the bottom right, showing 7 contributors.

Figure 1. (1) List of ALL the files in the repository, here “club-iEEs”. (2) Summary description of the latest commit (latest modification). (3) Date of the last commit. The latest commit will always be displayed above, in (4) to access the latest commit. Click on (5) to see the history of all previous commits. (6) General project information and (7) contributors.

To access the whole file history, you need to click on the file name (in (1)). To access only the changes done in one specific commit, click the text in (2).

Understand the README file

- The text is formatted with some markdown.

Open/download a file

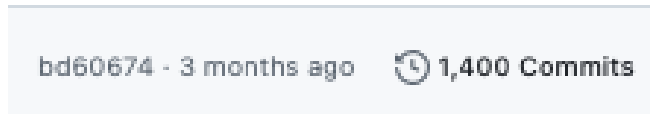
- Go to README.md and look inside the file (click on “raw”)
- Explore the folders and files of a repository
- go to the folder [/data](#) → which types of file can be stored in a repo?

1.5 Explore the history of changes in a repo

You can access to different aspects of the history of a repository :

See the history of commits

In the area n°5 you see on the left the name of the last commit, the date, and the numbers of commits. To see all commits, click on “commits” → you see the history of commits



See all files affected by the same commit

- Access the files concerned by any commit

Go to history of commits (see above), then click on the name of a commit → you see all files that were changed during this particular commit.

- Access the files concerned by the *last* relevant commit
- Go to the root of the LakeEnsemblR repo, then in area n°2, click on the last commit
→ You see all files modified for that specific commit

See the history of each file separately

To do that, click on the name of a file in area n°1 of fig.1 then click on history (see fig. 2)

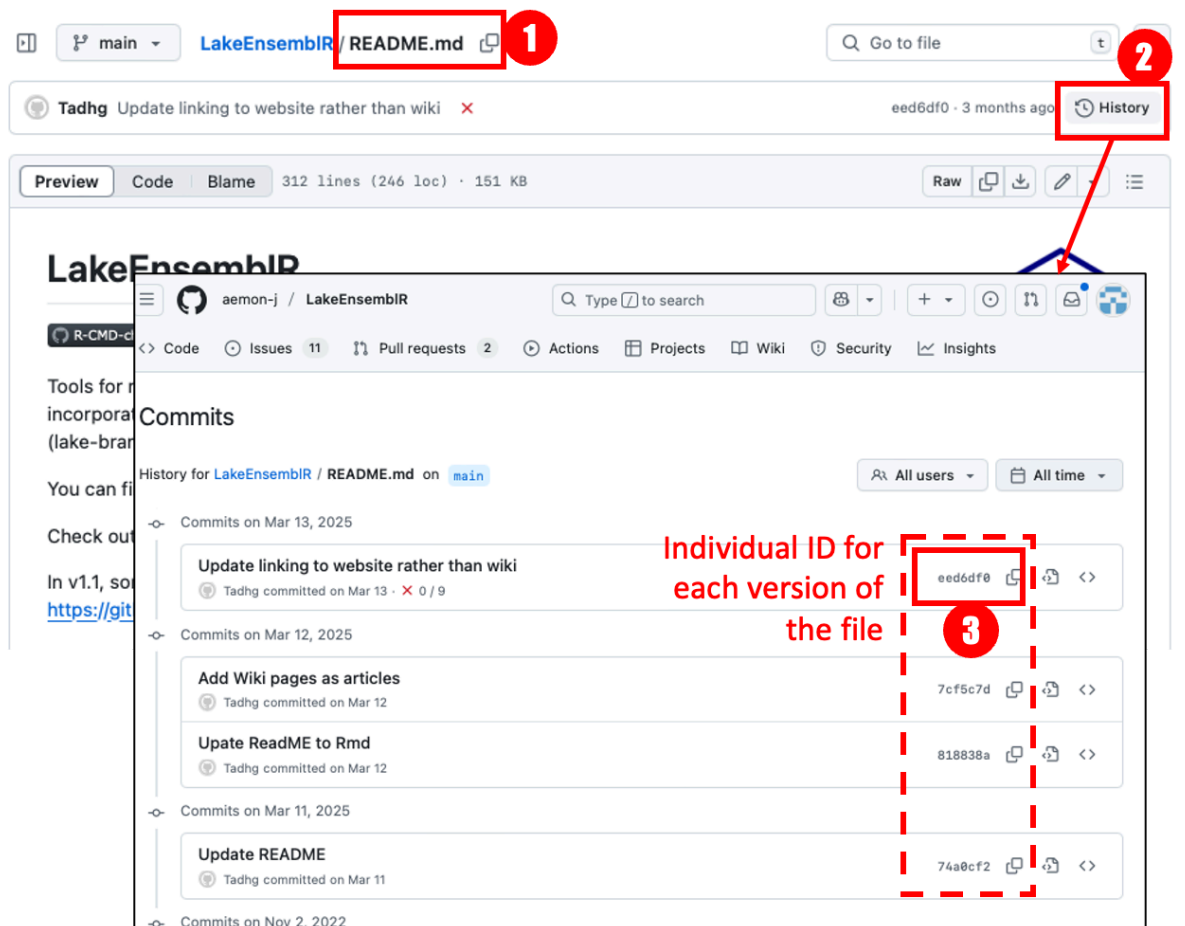





Figure 2. See one single file history. Click on a file name (see area (1) from Fig. 1), then History (2 in this figure). The window now shows all the different versions of your file. Click on (3) to see the diffs for a file (go to Figure 3).

See the different versions of a file

- Click on  to see the version of the file at a given point in history.
- Click on **Raw** to see the source of the file at this given point in history.
- You can do versioning with different file formats. For instance with excel. Demo :
- Go to the root of the repository of the club
- click on test.xlsx / history
- click on  of the version at the bottom (older version)
- click on “view raw” or “Download”

See the different versions of a repository

- Click on , you will see the whole repository in its state at a given point of history. **Warning:** the interface will look the same (that can be confusing!), but you are indeed browsing the repository as it was at that point in history.

See the detailed log of changes for a file (diff)

- Click on a version number (3 in Figure 2) (`c3c26e0`), and visualize the detailed log of changes (= diffs) (Figure 3).

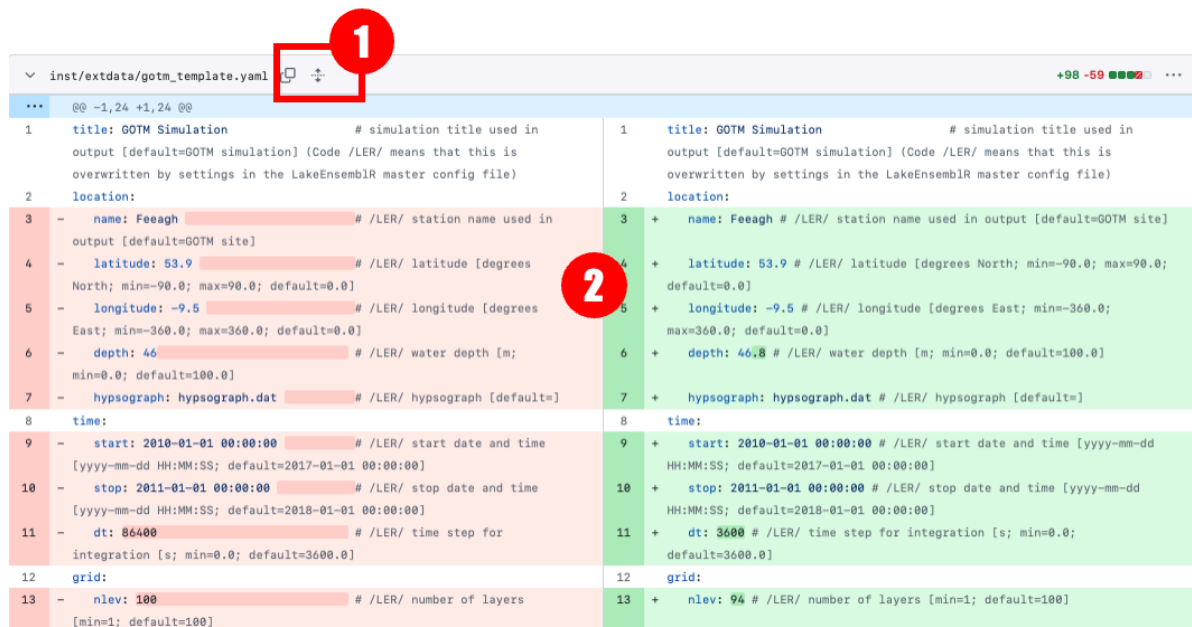


Figure 3. Visualize the differences between old and new versions. By default, only the lines with changes are shown, but click (1) to see the whole file. (2) Example of differences: some spaces were removed from the .yaml file.

II - Work with your own repositories

First, Install Git on your computer

Use Git with R studio : <https://r-pkgs.org/git.html#git-setup>

Use Git with GitHub Desktop : <https://desktop.github.com/>

(other resources are available, including GitHub for Atom <https://github.atom.io/>)

With R Studio

2.1° Create your own repository

<https://r-pkgs.org/git.html#git-init>

2.2° Do a local modification in the repo: “commit”

With R Studio : <https://r-pkgs.org/git.html#git-status>

2.3° send your modification on GitHub: “push”

With R Studio : <https://r-pkgs.org/git.html#github-init>

With GitHub Desktop

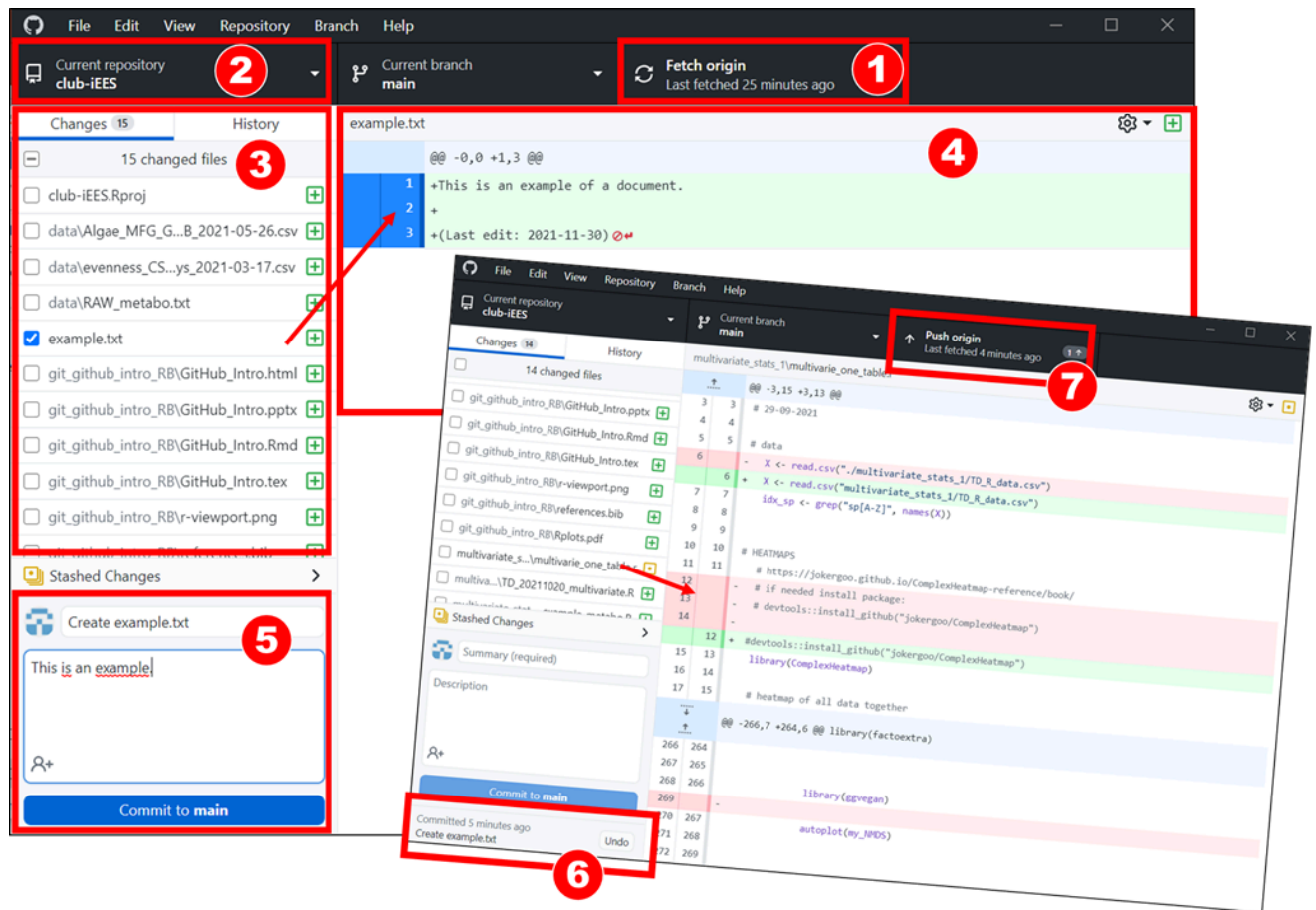
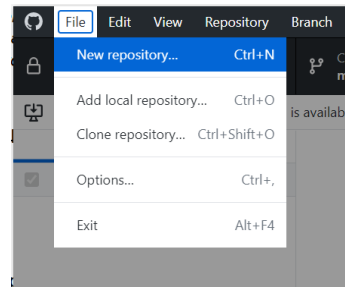


Figure 4. GitHub Desktop interface. (1) Before you start working, check you are up-to-date with the online repository by clicking “Fetch”. If there have been some changes, the icon will change to “Pull”. Click the same button again. (2) “Current repository”: drop-down menu with all the repositories you cloned. (3) List of files with differences between your local version and the online version. When you select a file (here, “example.txt”, it appears on the right side (4). The additions appear in green and the deletion appears in red. If you want to commit your changes, check the boxes near the file names in (3), then fill-in the commit field (5). You need at the very list a summary. Click “Commit to main” when you are ready. You can undo the commit while it hasn’t been pushed (6). Finally, once you are ready, click (7) to push your commit to the online repository.

2.1° Create your own repository

Option 1 : Create your own git folder in your computer and publish it to GitHub

1° File/New repository
(do not put it in a
synchronised folder such
as a dropbox folder)

A screenshot of the 'Create a new repository' dialog box in GitHub Desktop. The 'Name' field contains 'my_first_repo'. The 'Local path' field shows 'D:\'. The checkbox 'Initialize this repository with a README' is checked. The 'Git ignore' and 'License' dropdowns are both set to 'None'. At the bottom, there are 'Create repository' and 'Cancel' buttons.

2° Dialog box

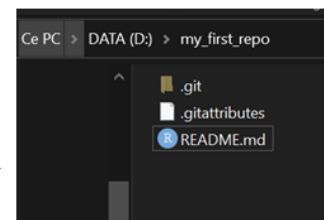
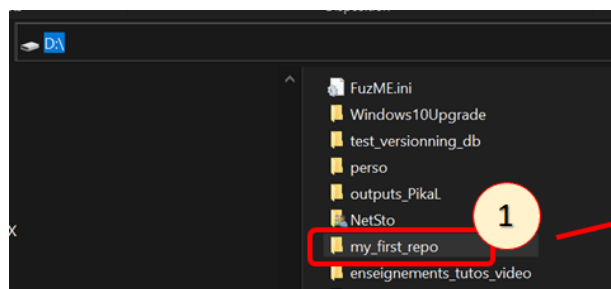
Name = name of the folder

Path = root of the folder

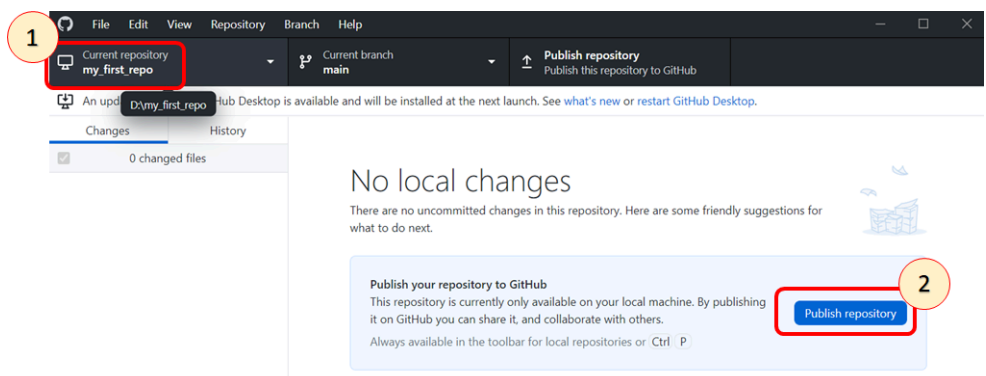
Initialize this repository with a README
creates a readme

3° You should have a new repo (1)

In your PC :



In GitHub Desktop



4) To Publish your
repo online,

click on Publish repository (2) → You're done !

Option 2 : create on Github.com then add it to your computer

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).

Required fields are marked with an asterisk (*).

Owner * / Repository name * 1

✓ myrepo is available.

Great repository names are short and memorable. Need inspiration? How about [improved-octo-funicular](#) ?

Description (optional)

2 ☒ Public
Anyone on the internet can see this repository. You choose who can commit.

☐ Private
You choose who can see and commit to this repository.

3 Initialize this repository with:

☒ Add a README file
This is where you can write a long description for your project. [Learn more about READMEs](#).

4 Add .gitignore
.gitignore template: R
Choose which files not to track from a list of templates. [Learn more about ignoring files](#).

4 Choose a license
License: MIT License
A license tells others what they can and can't do with your code. [Learn more about licenses](#).

This will set `main` as the default branch. Change the default name in your [settings](#).

Figure 5. Create a repository directly from GitHub. (1) Choose a name, (2) decide on the accessibility, (3) add a read me file, (4) Add a .gitignore file, (5) choose a license. (3-5) are optional but good practices.

Note that in GitLab you can create projects, and then repo that goes in the project. You can then grant automatic access to some group of users.. Can be really powerful in labs or within research teams.

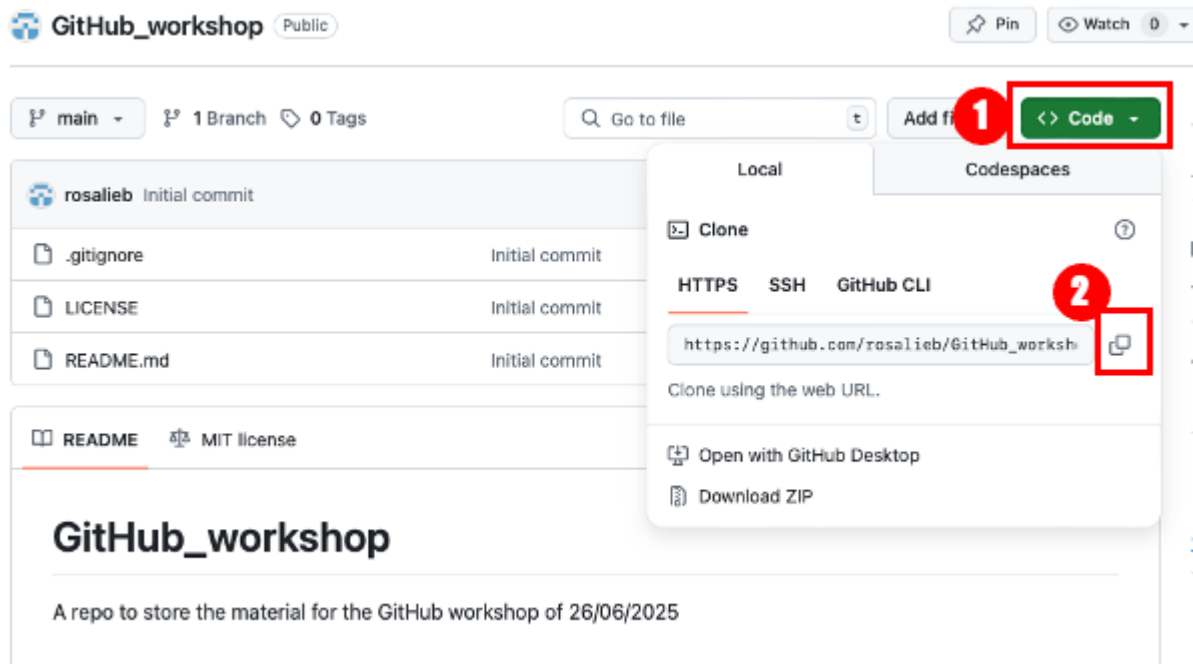


Figure 6. To add the repository locally, you can then click on code (1), and copy the link (2).

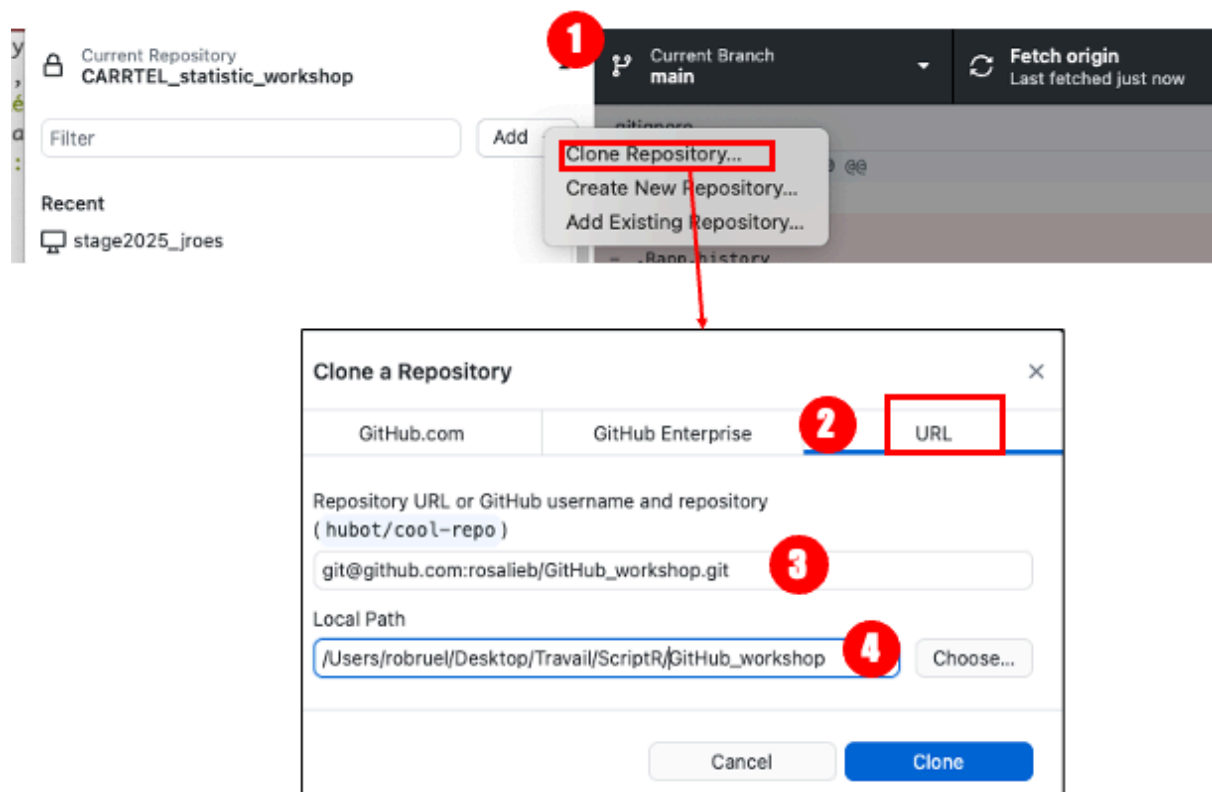
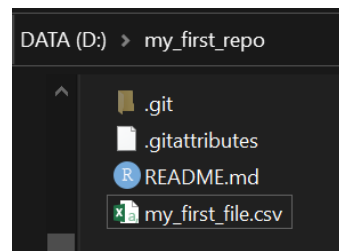


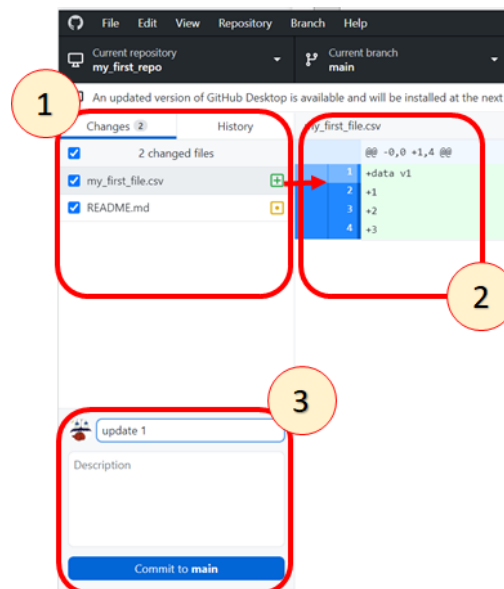
Figure 7. Open GitHub desktop, go to the top-left corner and click “Add” > Clone repository (1). A dialog box opens; select URL (2) and paste the link (3). You can select in (4) where to save this folder locally.

2.2° Do a local modification in the repository: “commit”

Let's say we modify the readme file (with a text editor) and add a csv file “my_first_file.csv”



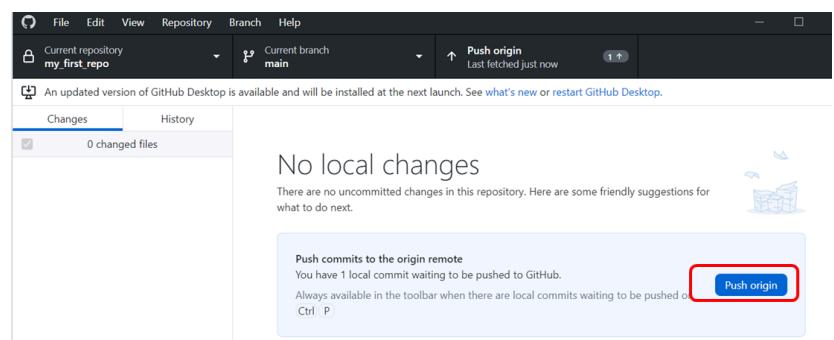
GitHub Desktop should automatically detect the files that has been changed (1)
If you click on a file, the changes (the “diffs”) will be displayed on side of it (2)



In order to save these modifications into the versioning system in your computer (nothing happens online so far), you need to “commit” this new version to the Git System (3)
You need to write a title of the commit and click “**Commit to main**” (3)

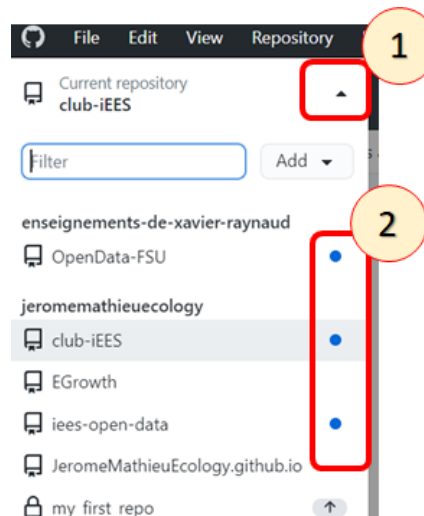
2.3° Send your modifications on GitHub.com : “push”

So far your modifications happened only in your computer.
To send the modifications online to GitHub.com, click “**Push Origin**”
This should send the edits to the online folder



2.4° Check if any local repository has a commit waiting to be pushed

In GitHub Desktop, click on the triangle on the side of the current repository (1)
All folders with changes that need to be pushed will be marked by a circle (2).



2.5° Compare you local folder to the online version and update you local folder

- Select your repository like in 2.4
- Click on Fetch Origin

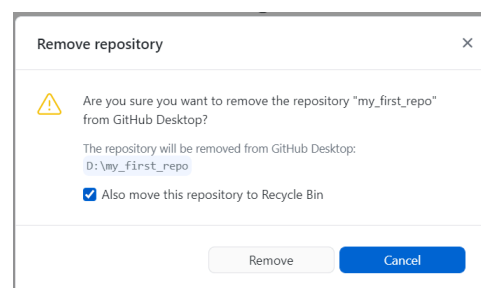
Any modification that is online but not on you computer should appear on the right part of the GitHub user interface.

- Click on pull to propagate the changes to your local computer. If there is a versioning issue, a message will appear.

2.6 Remove / delete a folder

Go to Repository/remove

You can just remove the folder from the git system (the folder stays in your pc),
or you can completely delete the folder by checking
“delete the folder”.



Exercices

You can do them with R studio or GitHub Desktop. The explanations for each system are given below.

- 1° Make a new Git repository in your computer, with a csv file.
- 2° Publish your repository on Github.com and check online that it is updated
- 3° Make/update in the readme with a text containing two levels of titles (hint : use # and ##) and italic words (hint : use *xxx*). Make changes in the csv file.
- 4° Commit and push to GitHub.com
- 5° Check the update on GitHub.com (F5 to refresh the webpage)
- 6° Go to the settings of the repository and explore them.
- 7° Go to Options and scroll down to the danger zone → you can make your repository public or private and you can delete your repo.

III - Work with others

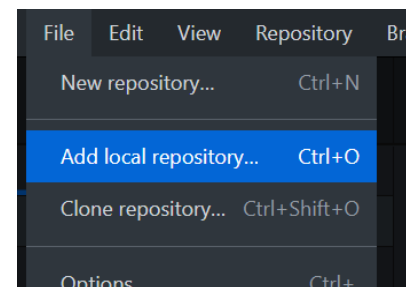
If you want to download an online repository, and eventually modify it, you first need to copy it to your computer. This is called “**cloning**” the folder.

To clone a repository:

- get the url of the repo (section I)
- go to File/ Add local repository
- then open URL tab and paste the URL
- If it is one of your own repo, it will listed

in the GitHub.com panel.

- Save it in a non synchronized folder (no dropbox/googledrive or other)



If you work on a repository that is administrated by someone else, your cloned version is called a “**branch**”. You can modify your branch and ask at some point to merge it to the original branch. This is called a “**pull request**”.

You can also make different versions of your repository in parallel. This is called a “**fork**”.

3.1° Visualize multiple branches - what your project may look like with multiple collaborators

From the home page of a project, click on Insight>Network and see the different branches for the same project.

Example: <https://github.com/JeromeMathieuEcology/club-iEES/network>

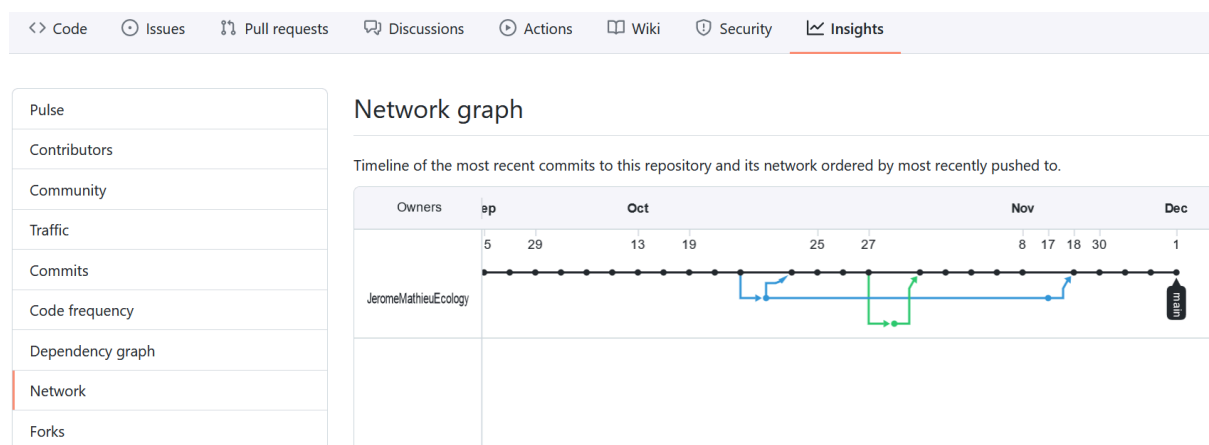


Figure 5. Network graph for the club-iEES, as of 01/12/2021.

3.2° Issue, fork and pull request

Issue = text message about a bug, a feature to add

Fork = separate version of the main repo developed by an Official Contributor, with specific modifications, that can be merged to the main branch, provided there are no conflicts.

Pull request = someone that is not an official Contributor, who cloned the repository in his own computer, modified it and wants to propose modifications to the repository. Only the contributors can accept or not the pull request.

3.3° Make a pull to a public repository

With R Studio :

With GitHub Desktop

3.4° Contribute to repository as a contributor

-
- Send to Rosalie (or another Contributor??) your GitHub ID → She will include you to a CARRETEL repo.
- clone the repo to your computer
- make a change to a file/ add a file
- commit the change (if you are 2+ people in the room, choose only one person to do that!)
- push the change [before being able to push the change, you need to “fetch” and “pull” any modifications from the online repository!]
- update the web page (F5) → you should see your changes!