FAIR_bioinfo for bioinformaticians

Introduction to the tools of reproducibility in bioinformatics

C. Hernandez[1] T. Denecker[1] J.Sellier[2] C. Toffano-Nioche[1]

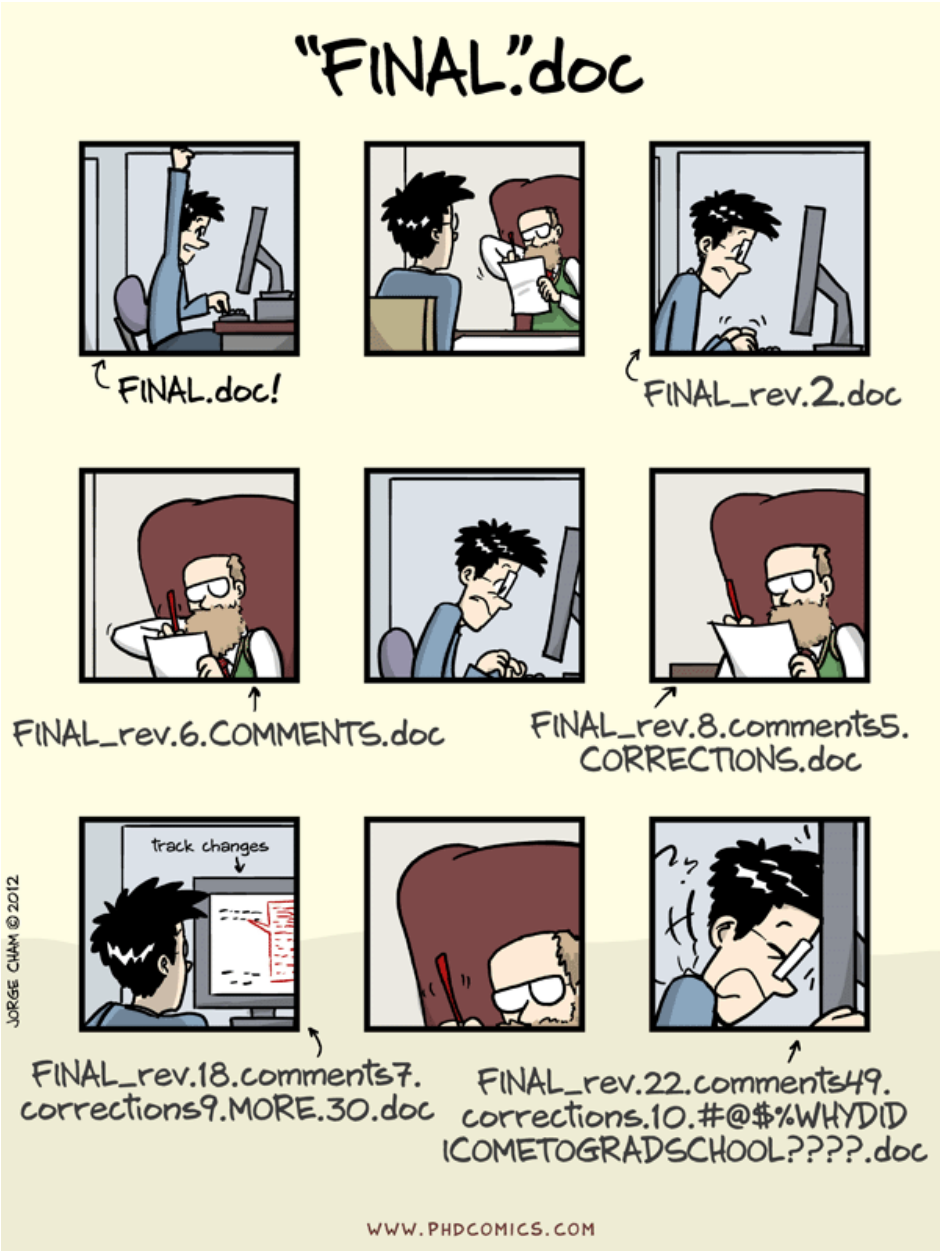
[1]Institute for Integrative Biology of the Cell (I2BC)
UMR 9198, Université Paris-Sud, CNRS, CEA
91190 - Gif-sur-Yvette, France

[2]Institut de Génétique et de Biologie Moléculaire et Cellulaire (IGBMC)
CNRS UMR 7104 - Inserm U 1258
67404 - Illkirch cedex, France

Sept. 2020

Introduction to code versioning

Really need of a files history?



"Most researchers are primarily collaborating with themselves," [Tracy] Teal explains. "So, we teach it from the perspective of being helpful to a 'future you'."

Files history = good practice for reproducible research

"Rule 4: Version Control All Custom Scripts"

OPEN ACCESS Freely available online

PLOS | COMPUTATIONAL BIOLOGY

Editorial

Ten Simple Rules for Reproducible Computational Research

Geir Kjetil Sandve[1,2]*, Anton Nekrutenko[3], James Taylor[4], Eivind Hovig[1,5,6]

1 Department of Informatics, University of Oslo, Blindern, Oslo, Norway, **2** Centre for Cancer Biomedicine, University of Oslo, Blindern, Oslo, Norway, **3** Department of Biochemistry and Molecular Biology and The Huck Institutes for the Life Sciences, Penn State University, University Park, Pennsylvania, United States of America, **4** Department of Biology and Department of Mathematics and Computer Science, Emory University, Atlanta, Georgia, United States of America, **5** Department of Tumor Biology, Institute for Cancer Research, The Norwegian Radium Hospital, Oslo University Hospital, Montebello, Oslo, Norway, **6** Institute for Medical Informatics, The Norwegian Radium Hospital, Oslo University Hospital, Montebello, Oslo, Norway

Replication is the cornerstone of a cumulative science [1]. However, new tools and technologies, massive amounts of data, interdisciplinary approaches, and

We further note that reproducibility is just as much about the habits that ensure reproducible research as the technologies that can make these processes efficient and

than to do it while underway). We believe that the rewards of reproducibility will compensate for the risk of having spent valuable time developing an annotated

Version control

Definition

version control, revision control, source control, or source code management: class of systems responsible for managing changes to files.

Feature

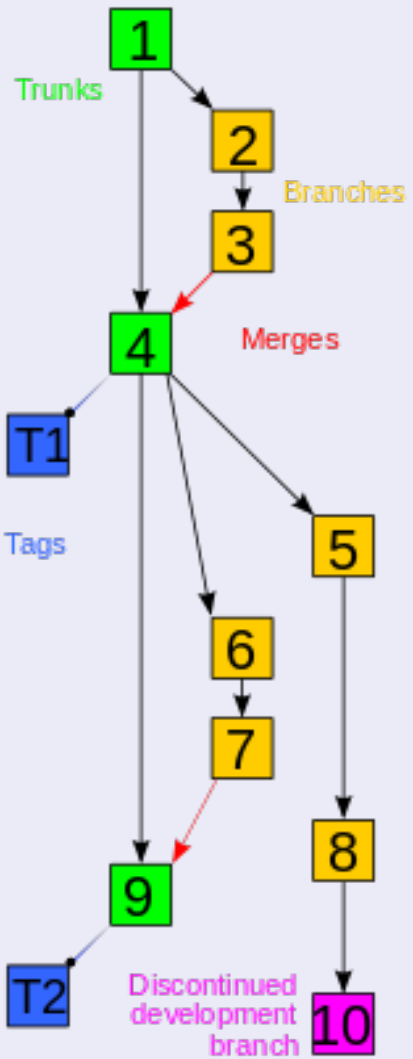
Each revision is associated with a timestamp and the person making the change.
Revisions can be compared, restored, and merged.

Software

SVN, Git, Mercurial, GNU arch, etc

wikipedia source

Revisions graph



Git and GitHub

Git

- will track and version your files
- enables you to collaborate with ... yourself
- open source license GPL (GNU General Public License)
- created in 2005 by Linus Torvalds for the development of the Linux kernel

GitHub

- stores your git repositories online
- enables you to collaborate with others (and yourself)
- first commit in 2007 by Chris Wanstrath, founded in feb. 2008, Microsoft Corporation still 2018

Git

Concepts, objects

- working directory: a user private copy of a whole repository of interest
- staging area: list of files of the working directory that will be considered for next commit (ie. could be not all the modified files)
- clone: a local copy of a repository (include all commits and branches), the original repository can be local, or remote (http access)
- commit: a git object, the snapshot of your entire repository compressed into a SHA (also the command the saves changes by creating the snapshot)
- HEAD: pointer representing your current working directory. Can be moved (git checkout) to different branches, tags, or commits
- branch: a lightweight movable pointer to a commit
- merge: combines remote tracking branches into current local branch

https://www.tutorialspoint.com/git/git_quick_guide.htm
https://www.powershellmagazine.com/2015/07/13/git-for-it-professionals-getting-started-2/

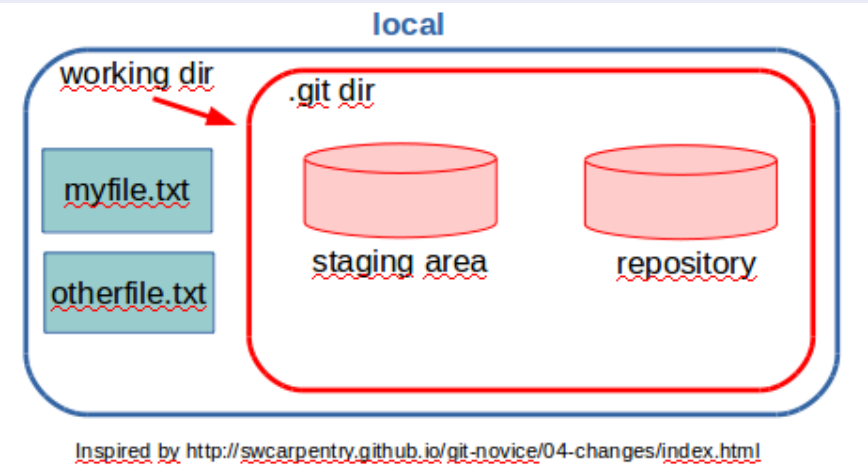
Setup

Git configuration: if not yet done, tell git our identity

```
git config --global user.name 'Your Name'
git config --global user.email 'Your Email'
```

Git repository initialisation

The initialisation (red arrow) is the creation of a .git repository:

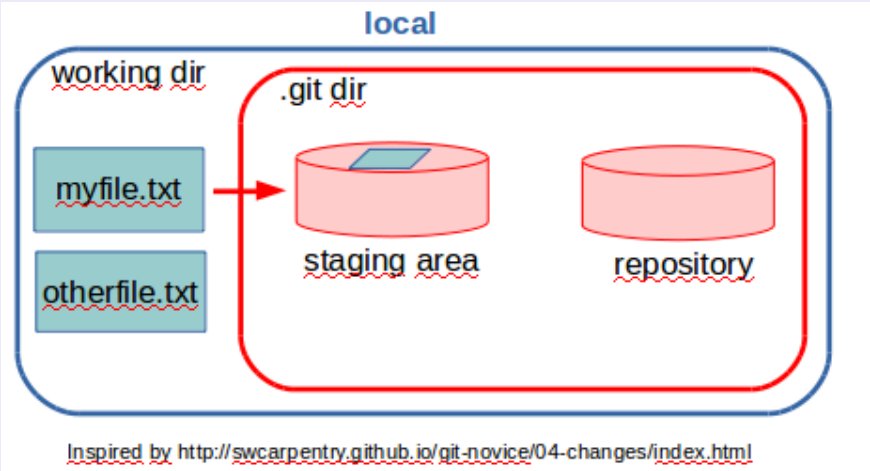


3 ways to initialize a .git repository:

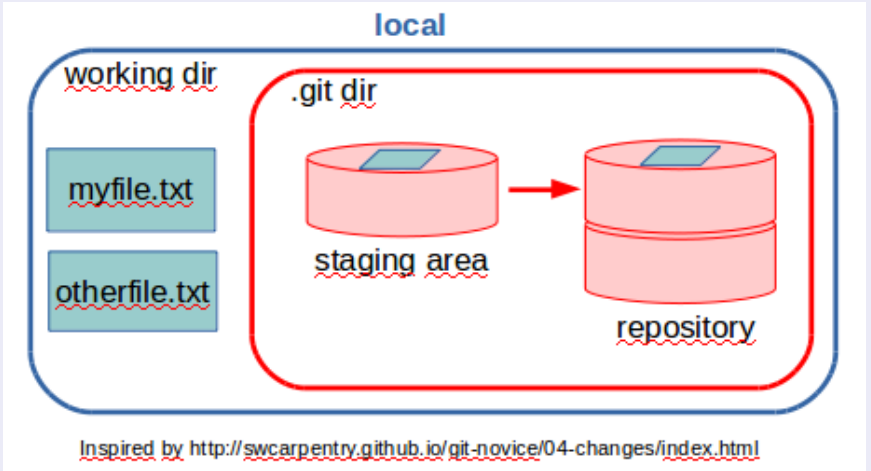
- git init: inside an existing folder (possibly containing files)
- git init project: create the folder "project" + initializes the .git subfolder inside it
- git clone /gitfolder/path /new/path copy the existing git repository to a new one

Tracking file

git add command for myfile.txt:



git commit -m "my reason":



`http://swcarpentry.github.io/git-novice/fig/git-staging-area.svg`

Git file states

Checking the file status: `git status`

File goes from untracked to tracked state (init), unstaged to staged state (add) and finally, to a committed state (commit).

Git Exercise

Objective

1st exercise

1. access and configure git
2. initialize a git repository
3. create files in this repository
4. use the basic git commands for tracking files changes (status, add, commit)

2nd exercise

5. copy another repository from github (clone)
6. use branching (branch) and merging (merge) to manage code changes

git setup: objectives 1 & 2

Git access by doker

```
docker run -i -t -v ${PWD}:/data continuumio/miniconda3
```

Git configuration

Global configuration (checking `user.name` with: `git config --list`):

```
git config --global user.name 'Your Name'
git config --global user.email 'Your Email'
```

Git repository initialization

On a new dedicated folder run:

```
git init # observe the .git folder (ls -la)
git status # find the current branch, "nothing to commit"
```

adding files: objective 3

create 2 files, check their git status: obj

```
for i in 1 2 ; do echo "file"${i}" text" > file${i}.txt ;
    done
git status # observe list of untracked files
```

add file1 to stating area

```
git add file1.txt
git status # observe the changing status of file1: untracked
     => staged
```

change file1 text

```
sed 's/text/text change/' file1.txt > tmp ; mv tmp file1.txt
git status # observe the 3 states, why file1 appears in "to
    be commited" and also in "not staged for commit"?
```

commit: objective 4

stage all files

```
1 git add file1.txt file2.txt # all files
2 git status
```

commit

```
1 git commit -m "1st commit + file1 change" # always add a
    message, use present time to explain the change
2 git status # all ok
```

So far, we have initiated a new project whose code is versioned by git: we have created files and all their successive changes were saved thanks to git.

We will now create a 2nd project by copying an already existing one. We're going to bring this project from an online git project site, *e.g.* github.

cloning: objective 5

copy of a project: clone

To download a project from github, we use the git `clone` command:

```
1 git clone https://github.com/clairetn/FAIR_bioinfo_github.
    git
```

observe result

- a new folder has been created (check with the shell `ls` command)
- its name is directly deduced from the url used
- this `FAIR_bioinfo_github` folder contains a `.git` repository and also a `README.md` file (see with `ls -la FAIR_bioinfo_github/`)
- it is a minimal project!

branching: objective 6

We plan to change the README file by adding our firstname at the authors list. With a git versioning system, a good practice is to create a branch to reserve the initial code until we validate our change.

create a branch nammed "branch1"

```
cd FAIR_bioinfo_github
git branch branch1
```

list all branches

```
git branch # find the star
```

branching: objective 6

go into the new "branch1"

```
git checkout branch1
git branch # find the star
git status # find the branch
```

work inot branch: change a file and keep change

Edit the `README.md` file and add your firstname to the "Authors list"

```
git status # file README.md is modified
git add README.md ; git commit -m "add my firstname in
    branch1"
```

return to master branch

```
git checkout master
more README.md # Is README.md modified or initial version?
```

merging branch: objective 6

We have check that our change is valid, so we now plan to move it into the `master` branch.

merge branch, then delete branch

```
1 git merge branch1
2 more README.md # what README.md version?
3 git branch -d branch1 # -d for delete
```

GitHub

GitHub "Who" ?

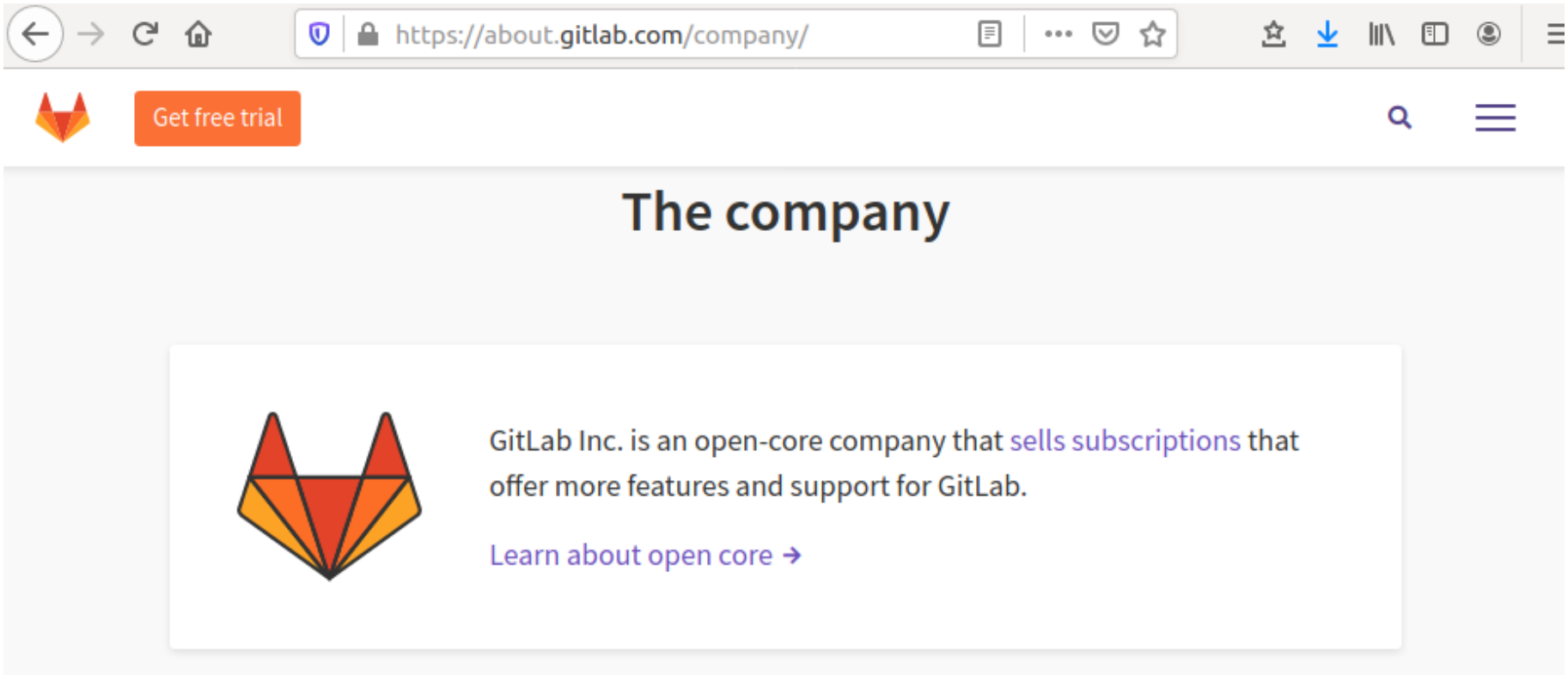
Quizz

1. public institute (governmental)?
2. semi-public institute?
3. not-for-profit organisation?
4. private company?

Response

See `https://github.com/about`: Careers' paragraph, you'll see a "company" word

GitLab, a GitHub alternative?



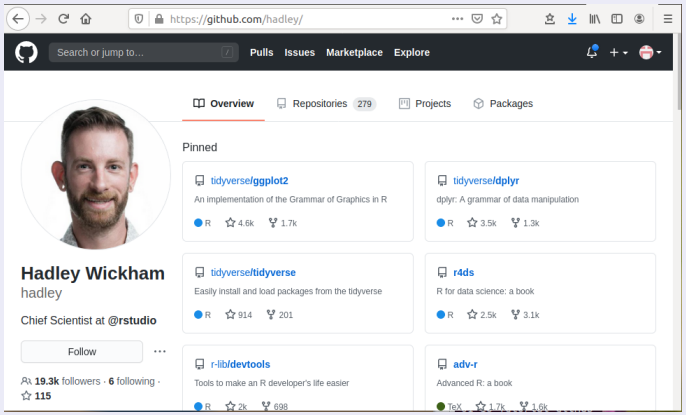
GitHub "What" ?

Quizz

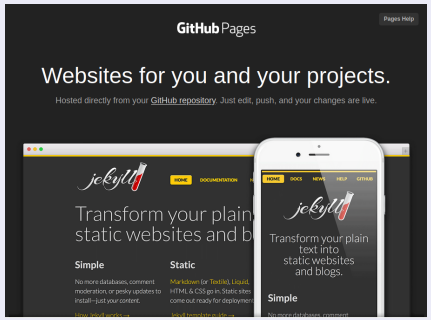
1. social network?
2. desktop application?
3. tool to create websites?
4. stable repository to publish any file?

GitHub

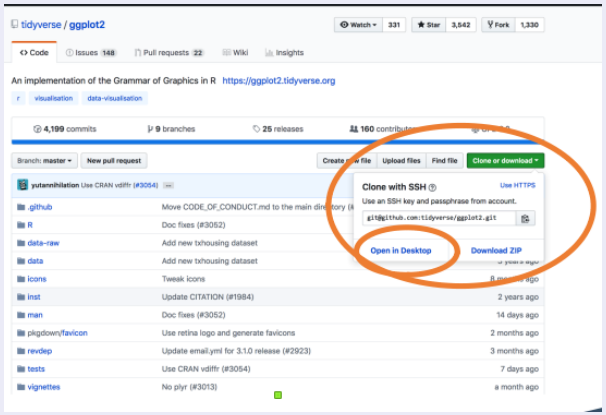
a social network ✅



a tool to create websites ✅



a desktop application ✅



a stable repository ...

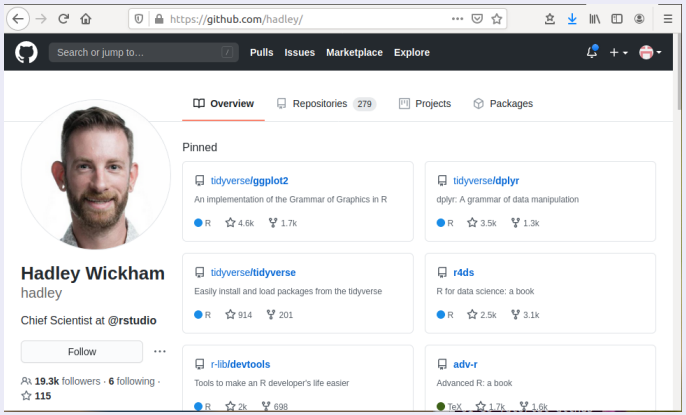
Popularity [edit]

Name	Users	Projects	Alexa rank (lower = more popular)
Assembla	Unknown	526,581+[46]	35,454 as of 28 July 2020[47]
Bitbucket	5,000,000[48]	Unknown	1,341 as of 28 July 2020[49]
Buddy	Unknown	Unknown	39,857 as of 28 July 2020[50]
CloudForge	Unknown	Unknown	402,884 as of 28 July 2020[51]
Gitea	Unknown	Unknown	236,332 as of 28 July 2020[52]
GitHub	31,000,000[53]	100,000,000[53]	78 as of 28 July 2020[54]
GitLab	100,000[55]	546,000[56][k]	2,710 as of 28 July 2020[57]
GNU Savannah	93,346[58]	3,848[58]	162,054 as of 28 July 2020[59]
Launchpad	3,965,288[60]	40,881[61]	11,533 as of 28 July 2020[62]
OSDN	54,826[63]	6,294[63]	8,708 as of 28 July 2020[64]
Ourproject.org	6,353[65]	1,846[65]	1,083,012 as of 28 July 2020[66]
OW2 Consortium	Unknown	Unknown	1,509,677 as of 28 July 2020[67]
Rosetta code	Unknown	Unknown	66,029 as of 28 July 2020[68]
SEUL	Unknown	Unknown	1,602,812 as of 28 July 2020[69]
SourceForge	3,700,000[70]	500,000[70]	470 as of 28 July 2020[71]
Name	Users	Projects	Alexa rank (lower = more popular)

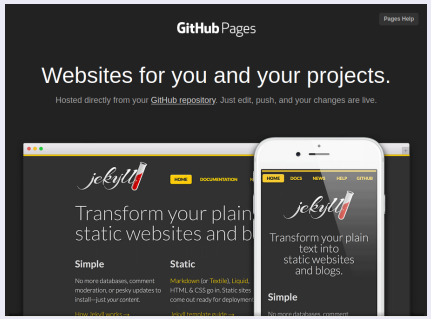
en.wikipedia, comparison of source-code-hosting facilities

GitHub

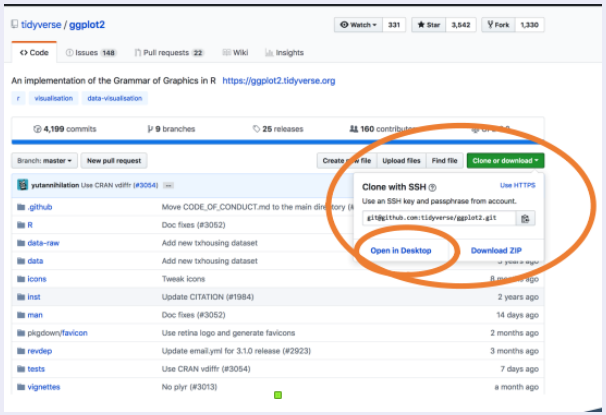
a social network ✅



a tool to create websites ✅



a desktop application ✅



... to publish any file ✅ ❌

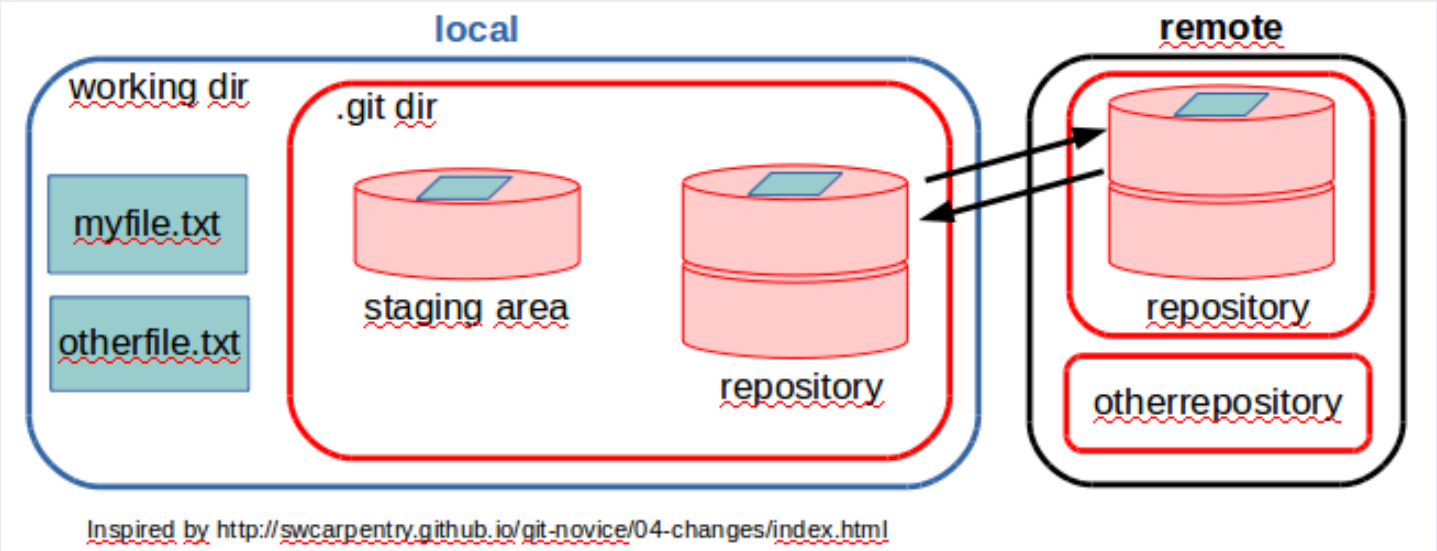
Files for which git can calculate the difference between versions.
Usually txt files of reasonable size:

- R script: ✅
- Python script: ✅
- pdf file: ❌
- fastq file: ❌

GitHub main usage: sharing code with others

GitHub:

- so used that Microsoft was interested in it (bought in june 2018)
- web-based: graphical interface + many more features than git
- git-based: git concepts and commands are retained
- commands for "sharing": `git push origin master` (local to remote) and `git pull origin master` (remote to local):



Concepts, objects

- user: your account on GitHub (unlimited for academics)
- organization: account for one or more user (e.g., swcarpentry)
- local GitHub: copies of GitHub files located your computer
- remote GitHub: your GitHub files located on https://github.com
- fork: a copy of a GitHub repository to your own GitHub account
- push: send changes on the working repository to your remote GitHub repository
- pull: copy changes on the remote GitHub repository to your local GitHub repository (useful when multiple people make changes)
- pull request: propose your changes to the initial forked GitHub repository. Also a place to compare and discuss the differences introduced on a branch with reviews, comments, integrated tests, etc

GitHub Copy a project

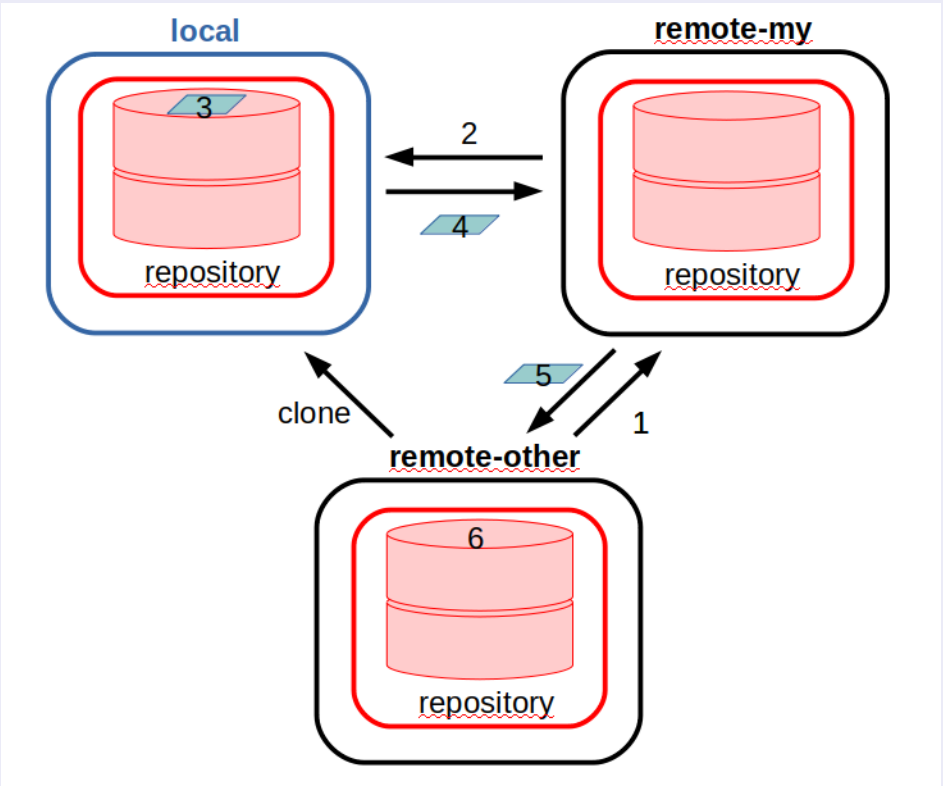
Clone *vs.* Fork?

- clone is git, fork is github
- all 2 copy a .git repository: clone copy it in your local machine, fork in your github account (do a clone)
- good practice: work (change files) in the local copy, not in the github copy (only for minor changes)
- to share your changes with the original repository, need a fork (by the way of a pull request)

See here an historical point of view of those 2 words.

GitHub Copy a project

Recommended flow to collaborate



(direct clone from github don't allow to collaborate)

- 1: fork a repository of interest in your github account
- 2: clone from your github account to your local place
- 3: make change (branch, add, commit, merge)
- 4: push change to your github account
- 5: pull request to propose your change to the initial project
- 6: wait (discuss) for integrating your change or not

GitHub Exercise 1

GitHub

Objectives

The objective of this exercise is to propose change to an existing project. We will:

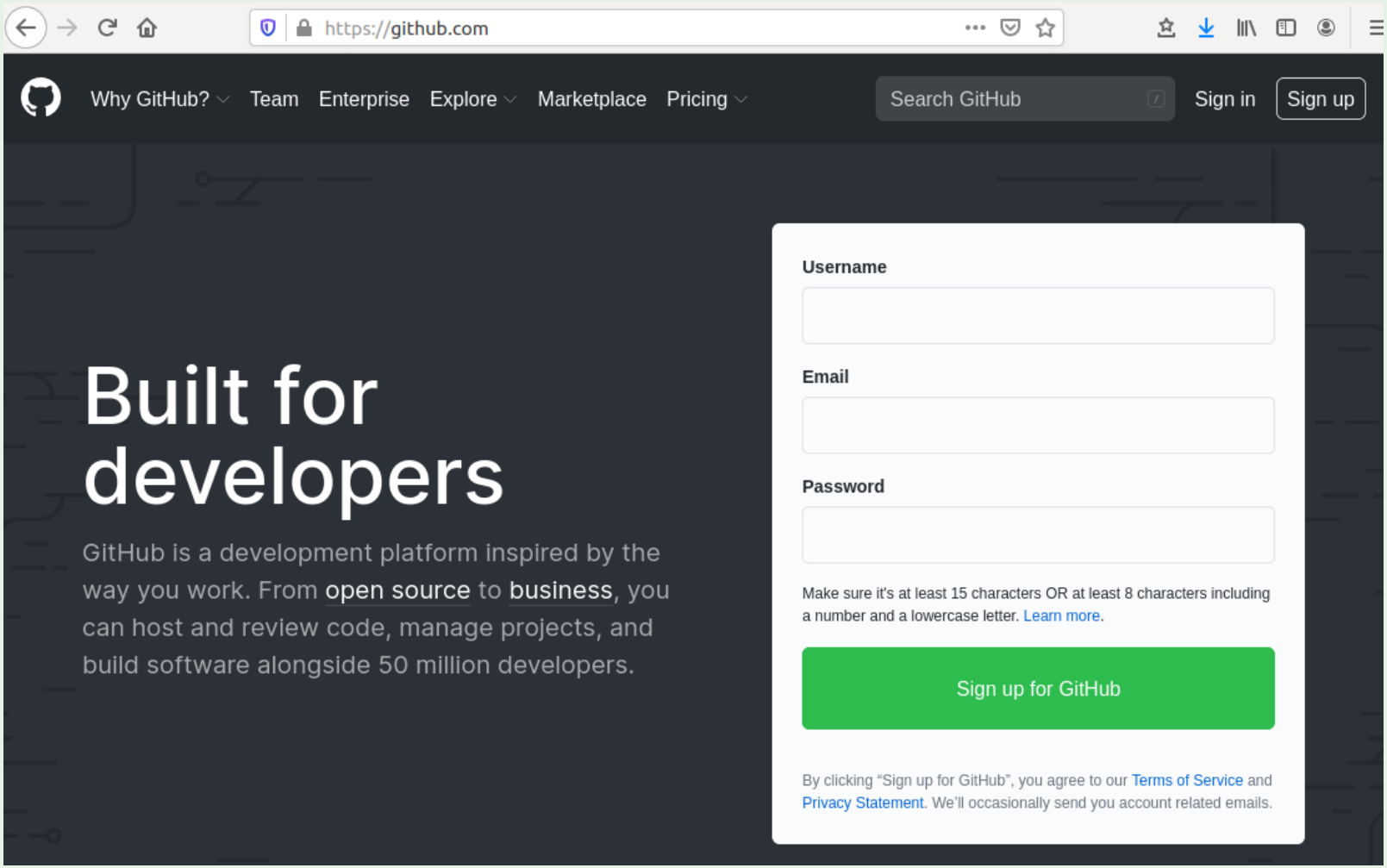
- fork an existing project to our GitHub account
- create a branch
- made a change in the branch
- save change into the change
- merge the branch

Web interface

During this exercise, most of the actions that will be performed will be done via the GitHub web interface, i.e. with many button clicks. The following pages will guide us to the next action.

GitHub: Access

With a browser, go to github (`https://github.com`). If not already yet, sign up and create your github account, otherwise sign in



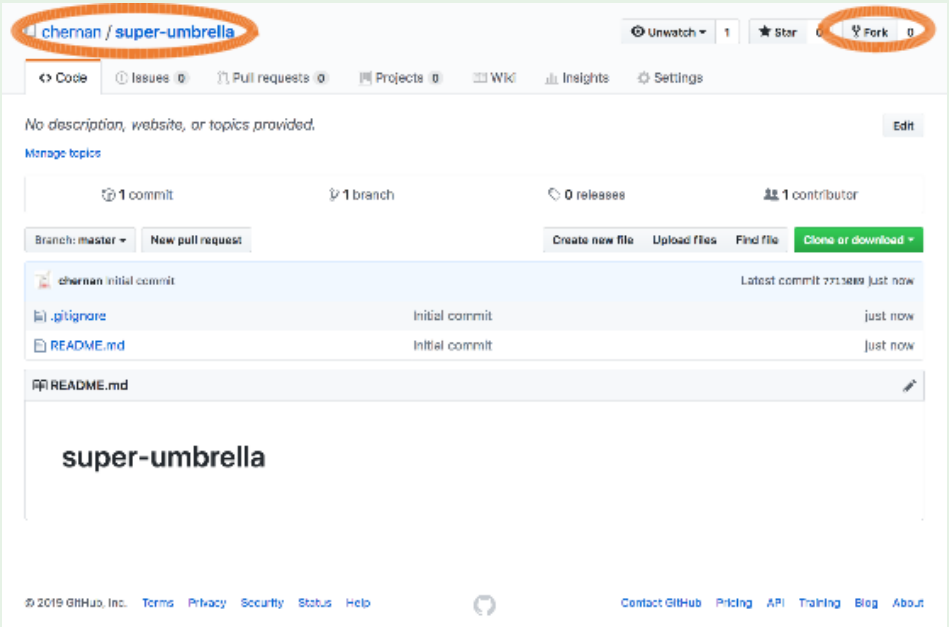
GitHub: fork a project

Objective

For this exercise, we will replay the addition of our first name, but by using the user interface proposed by github.

Fork in our gituhb account

With a browser, go to the url of the initial project,super-umbrella and click to "Fork" (upper right):

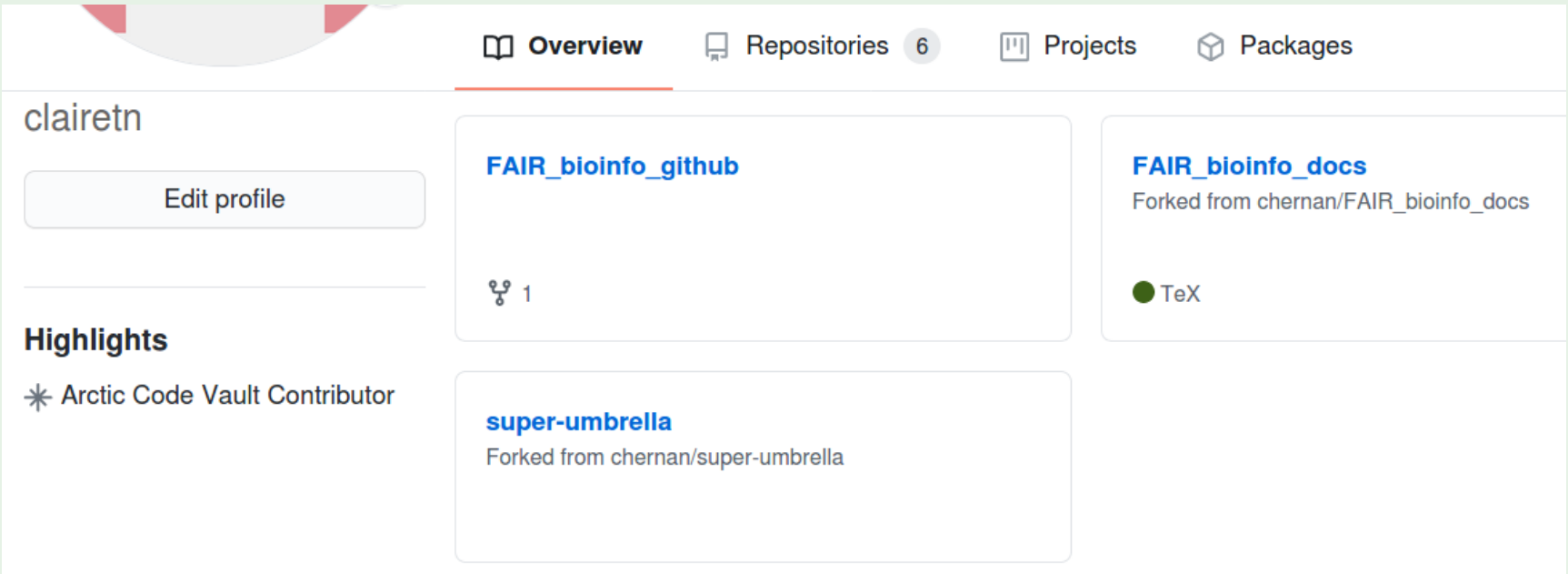


GitHub: the forked repository

Result:

You can see the result in your Github Overview: you have a new repository, named `FAIR_bioinfo_github` and entitled "forked from chernansuper-umbrella".

resultof the fork chernan, super-umbrella project:

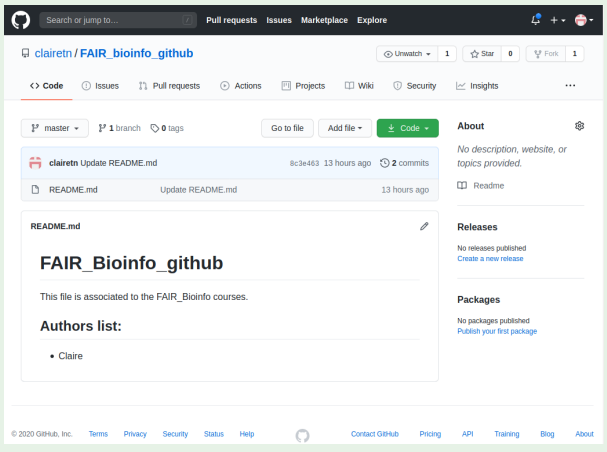


GitHub interface

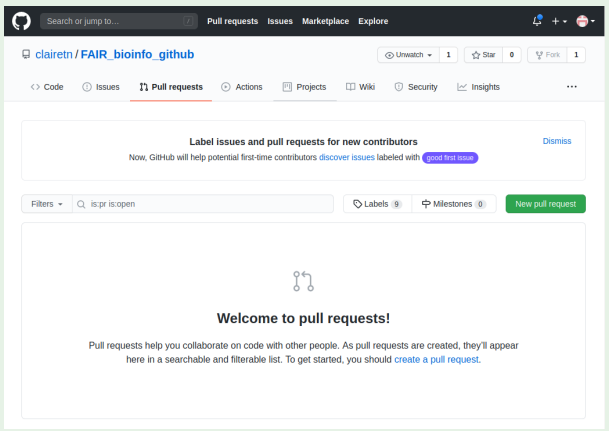
Tabs

9 Tabs offered by GitHub for each repository:
Code, Issues, Pull Requests, Actions, Projects, Wiki, Security, Insights, Settings.
Mainly focus on 3 of them:

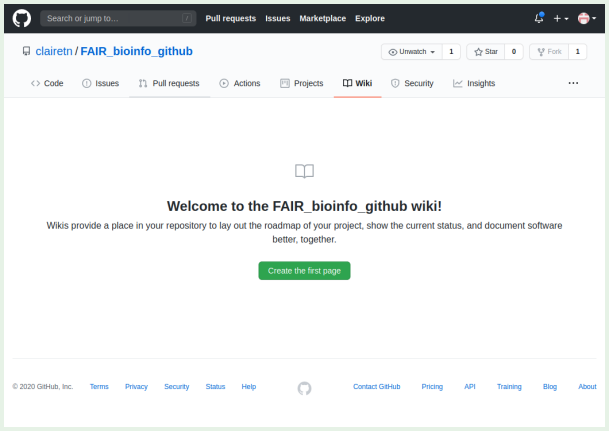
Code



Pull Requests



Wiki



GitHub

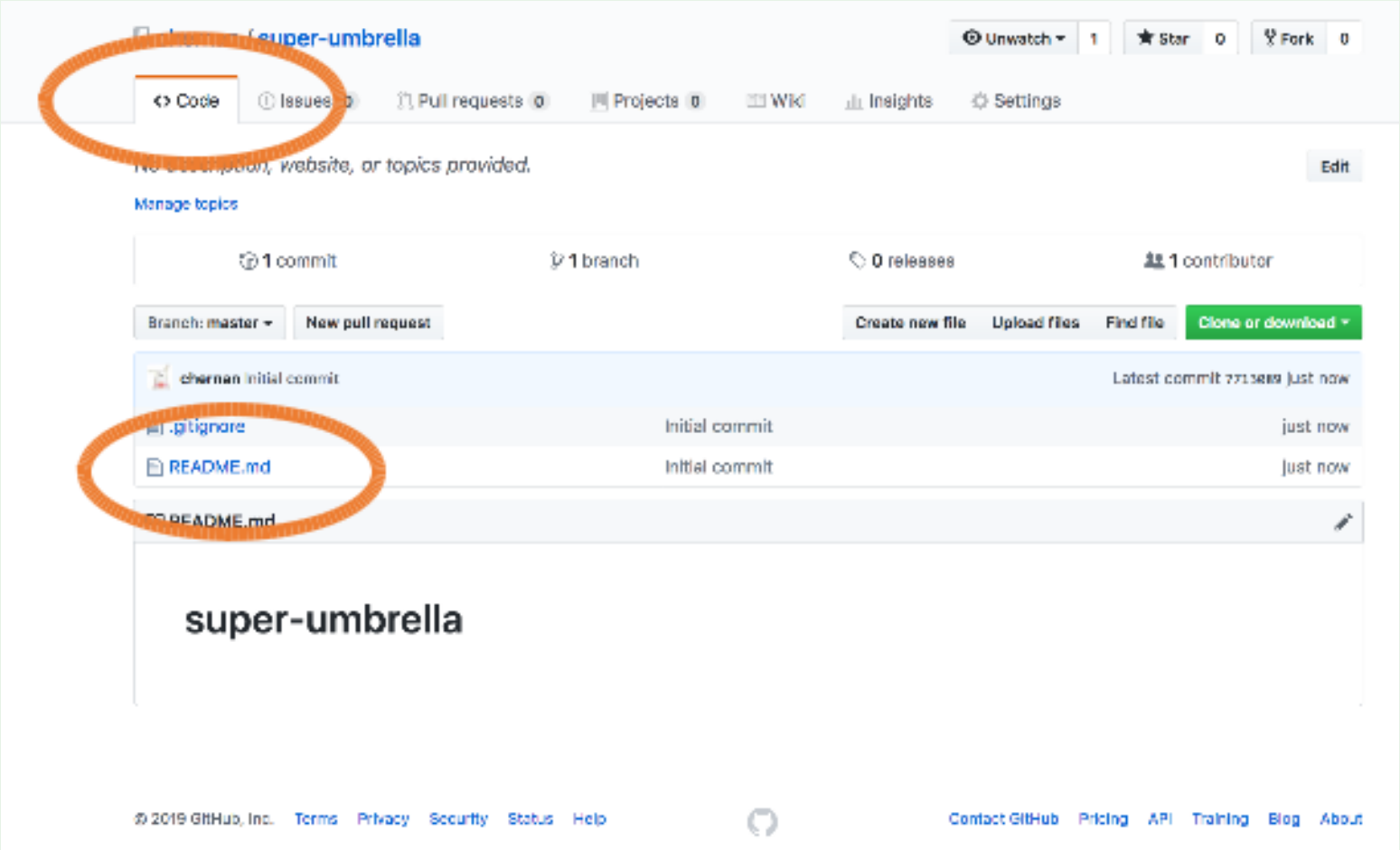
Previous exercises with git

- copy a github repo. (git clone)
- go to the local repo. (cd)
- create branch (git branch)
- go to branch (git checkout)
- make change (edit file)
- stage change (add)
- version change (commit)
- go to master (git checkout)
- merge branch (git merge)
- delete branch (git branch -d)

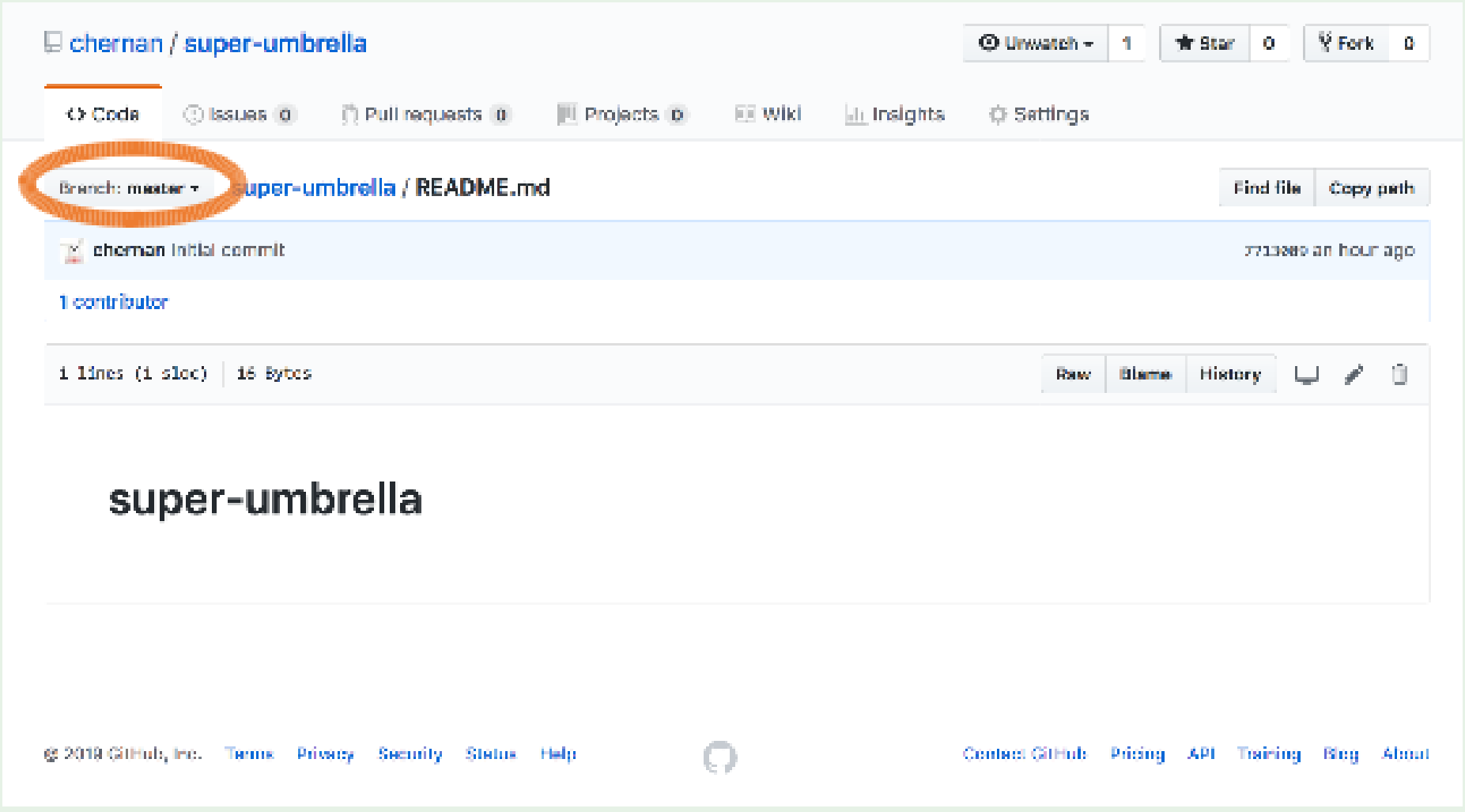
Next steps with github GUI:

1. fork a github repo. (just done)
2. create branch
3. make change (Edit file)
4. version change (commit)
5. compare branch to master
6. merge branch
7. ask for merging (Pull Request)
8. delete branch

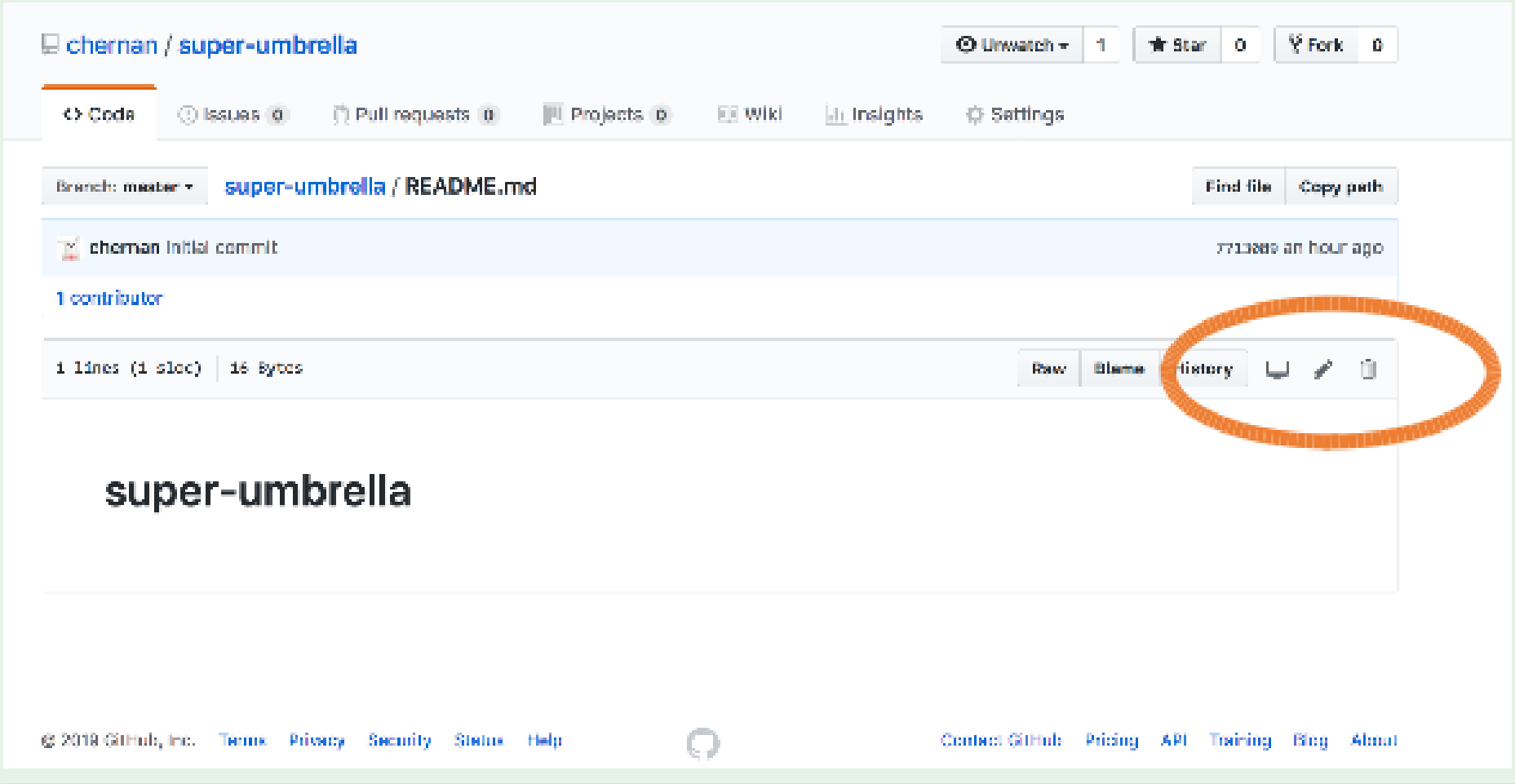
1: fork chernan, super-umbrella, see the README.md file:



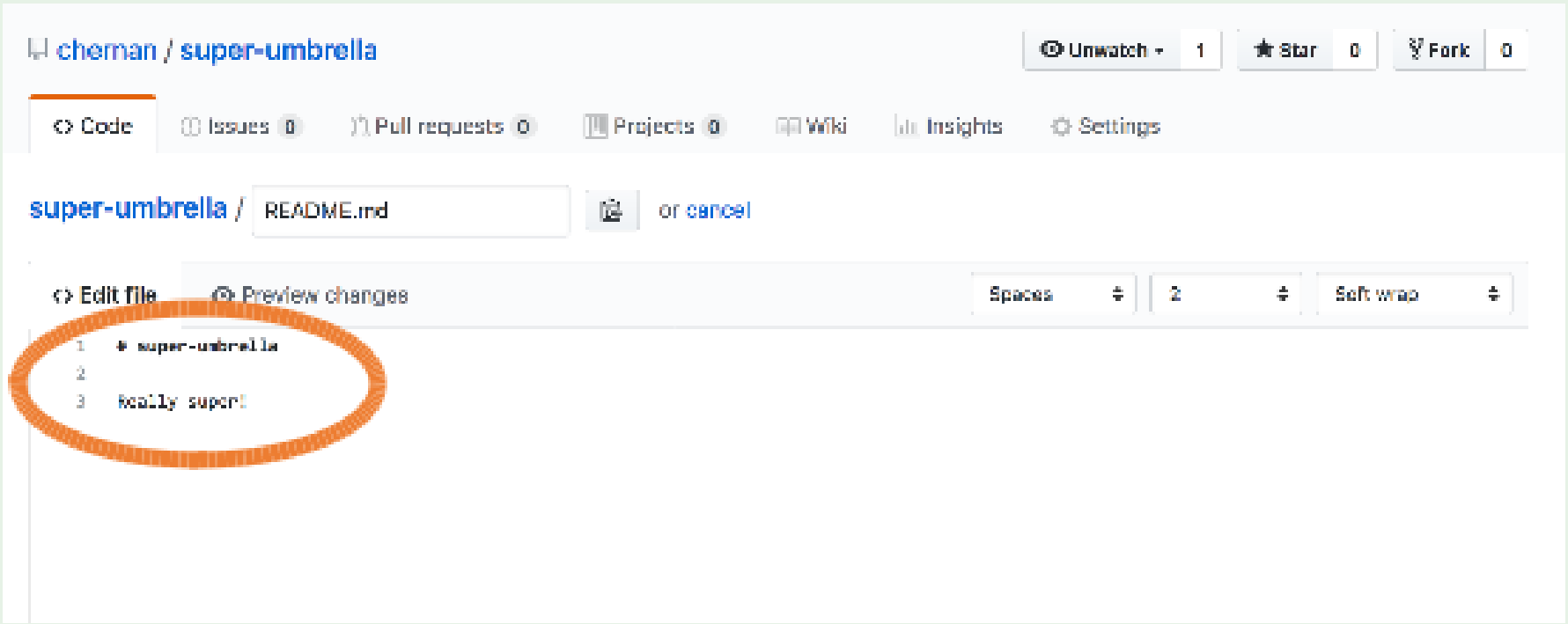
2: create a new branch, named "devel"



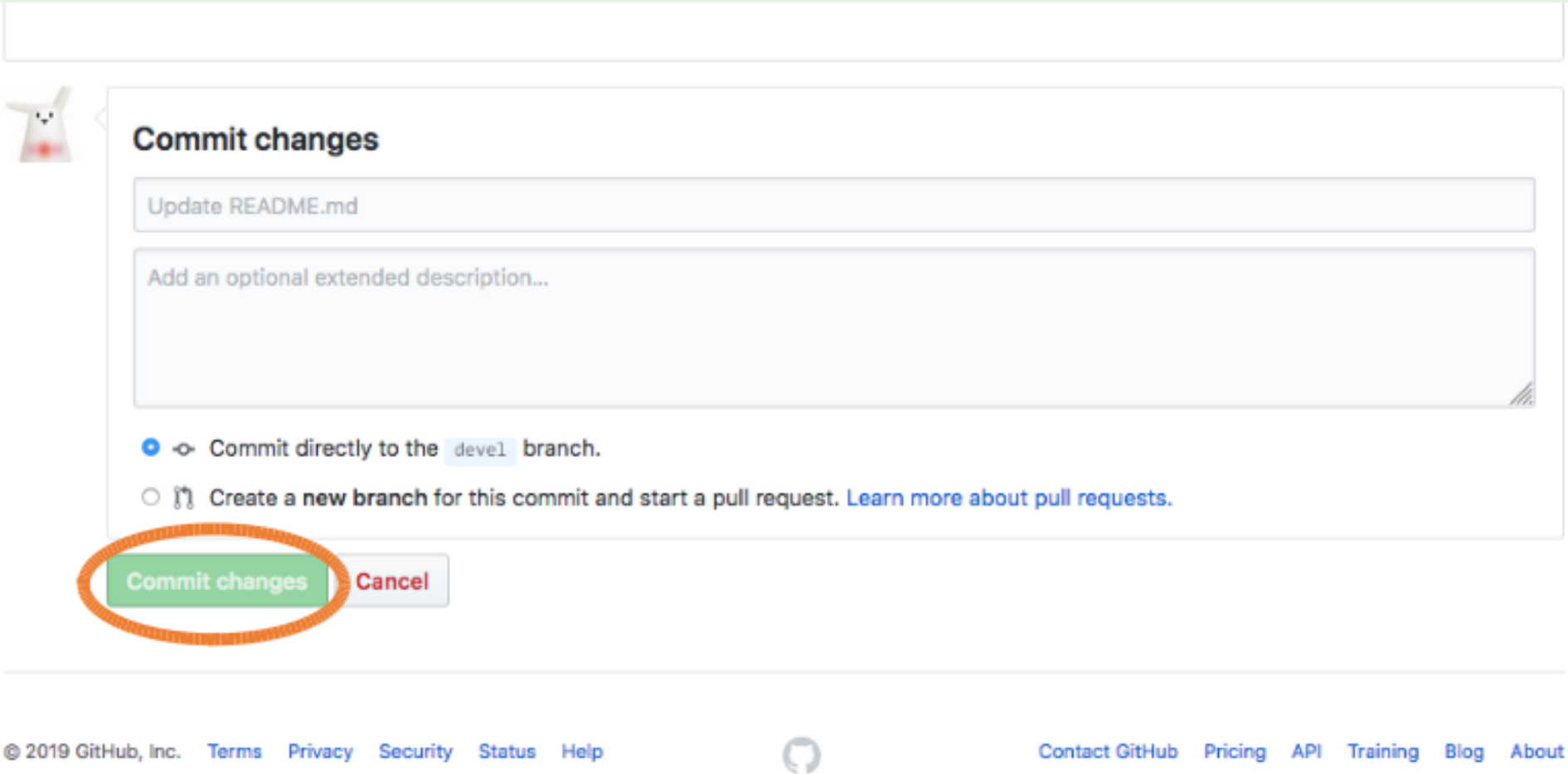
3: edit README.md to make change



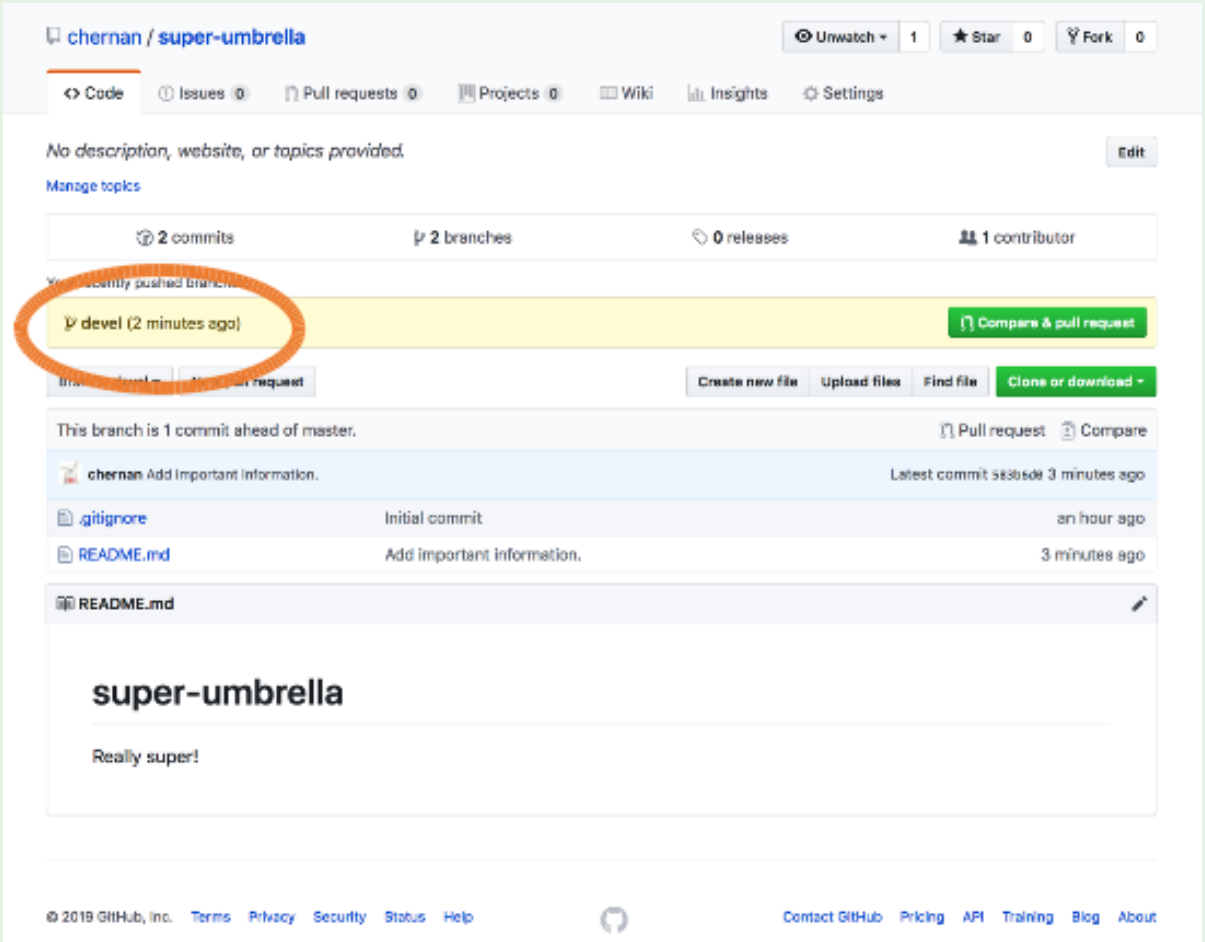
3: make change



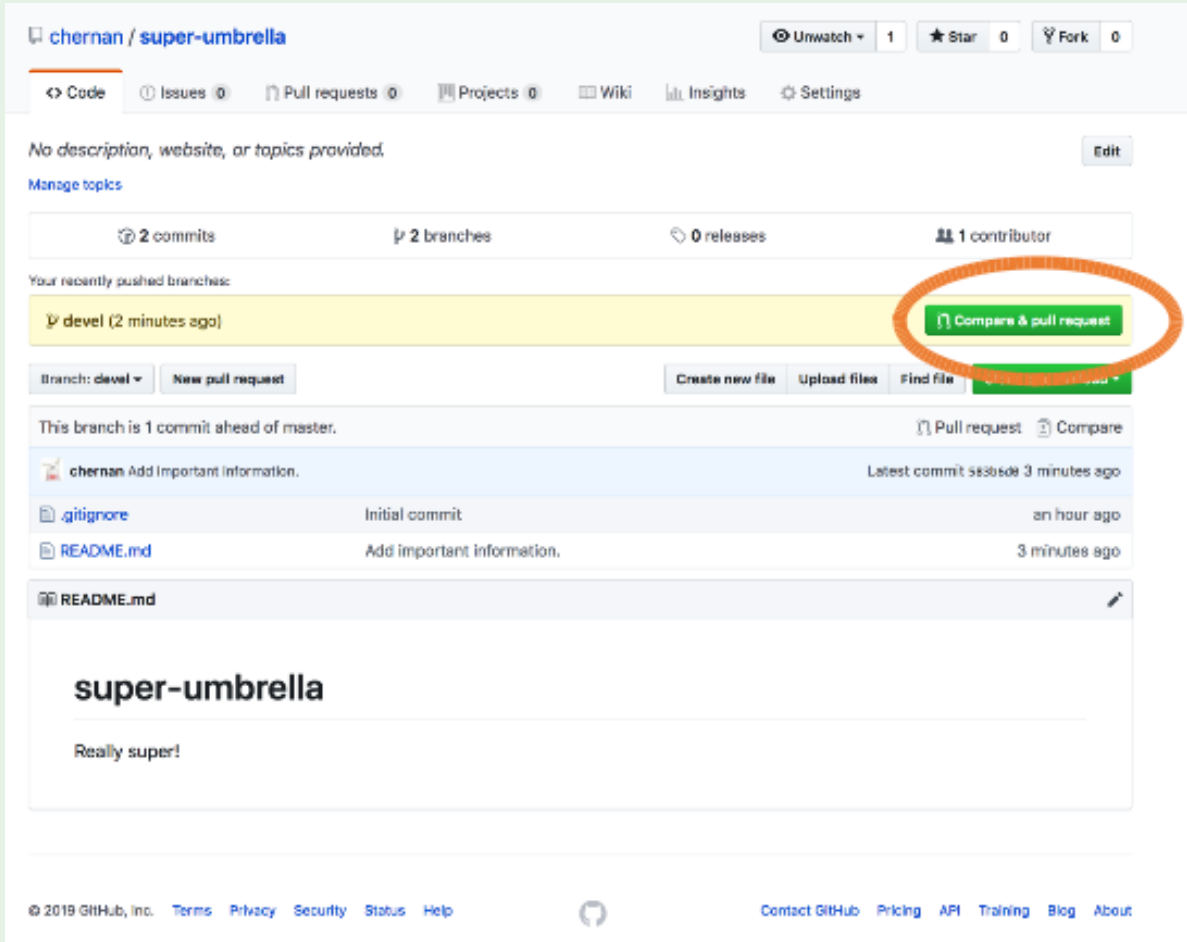
4: commit


Commit changes
Update README.md
Add an optional extended description...
Commit directly to the devel branch.
Create a new branch for this commit and start a pull request. Learn more about pull requests.
Commit changes
Cancel
© 2019 GitHub, Inc. Terms Privacy Security Status Help
Contact GitHub Pricing API Training Blog About

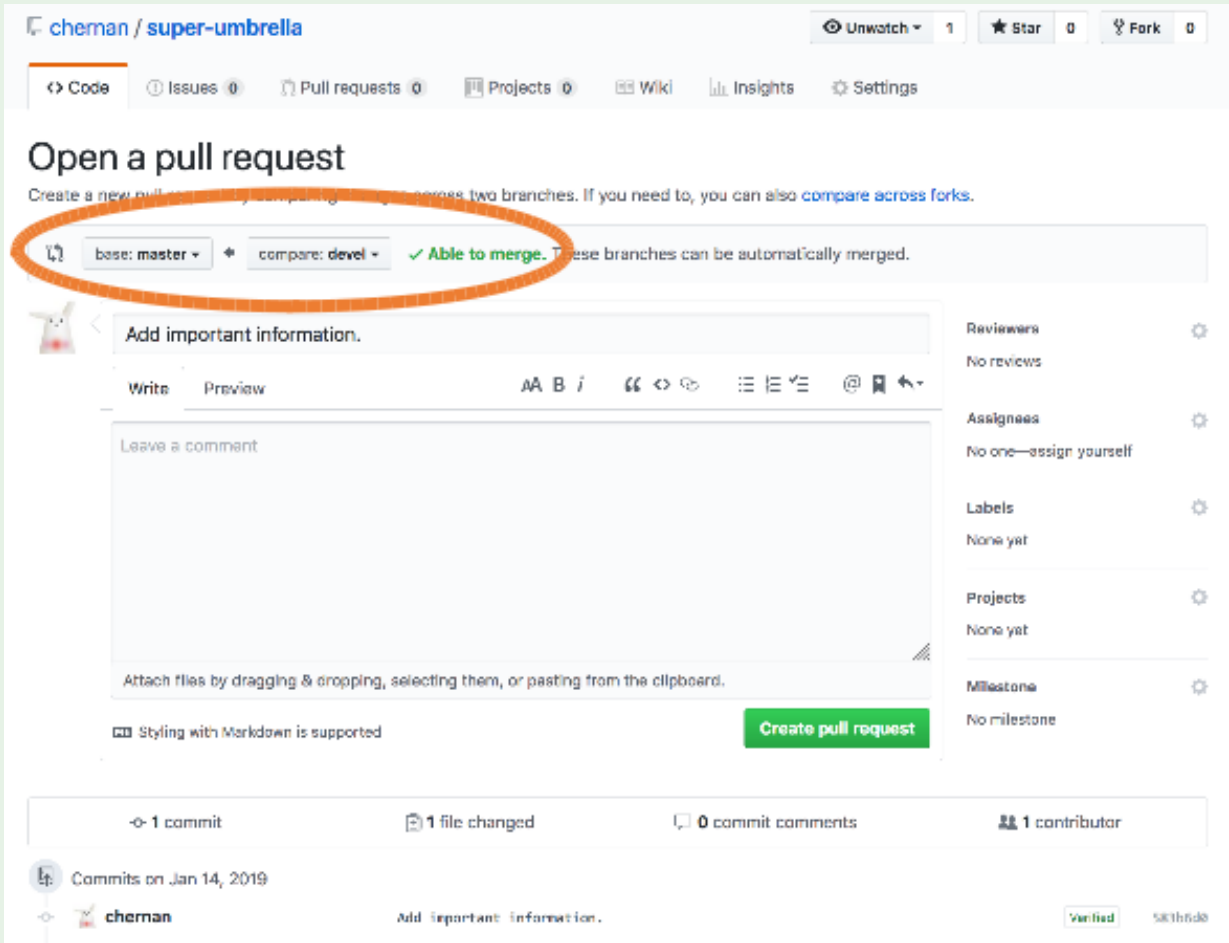
4: commit and pull request



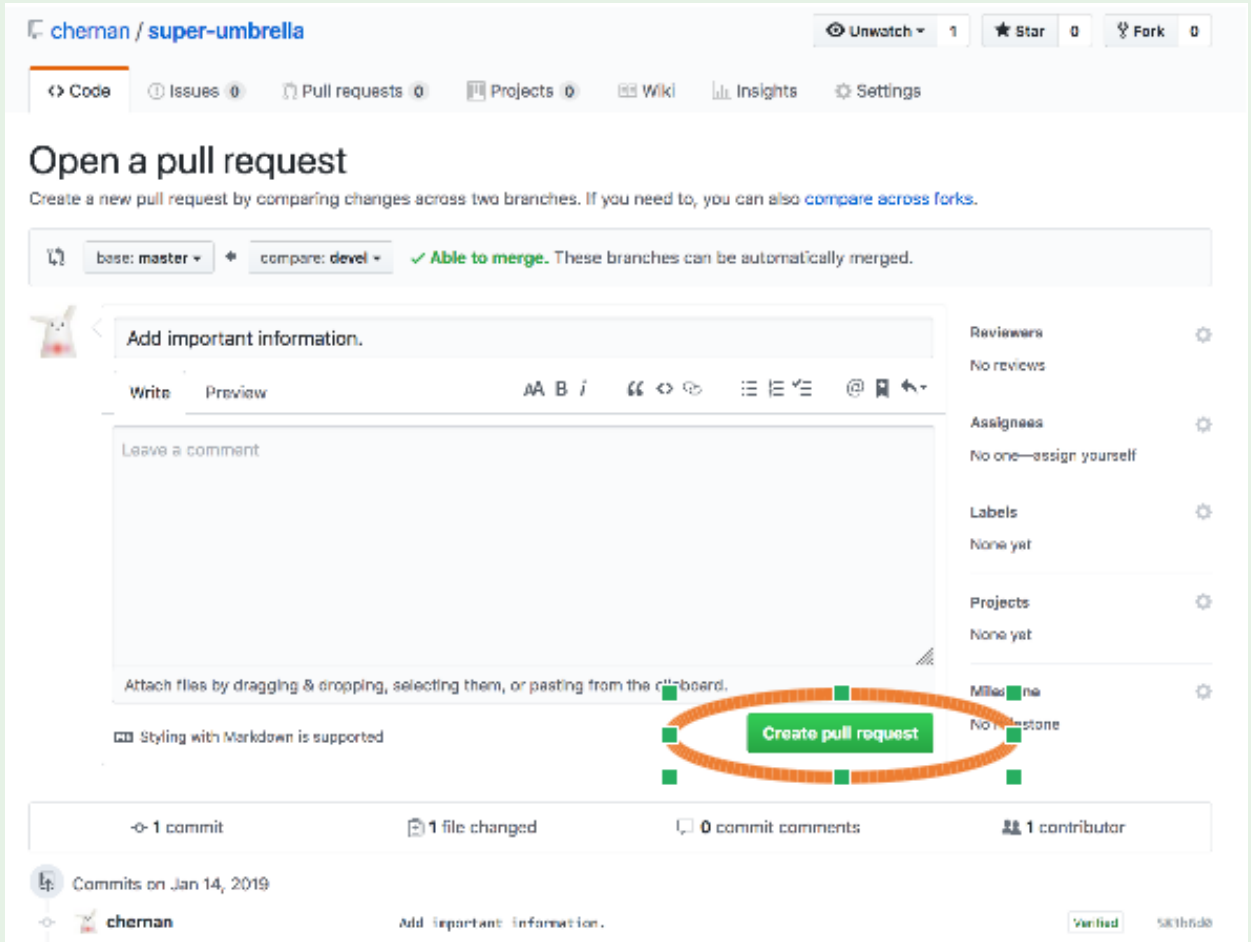
5: pull request, compare



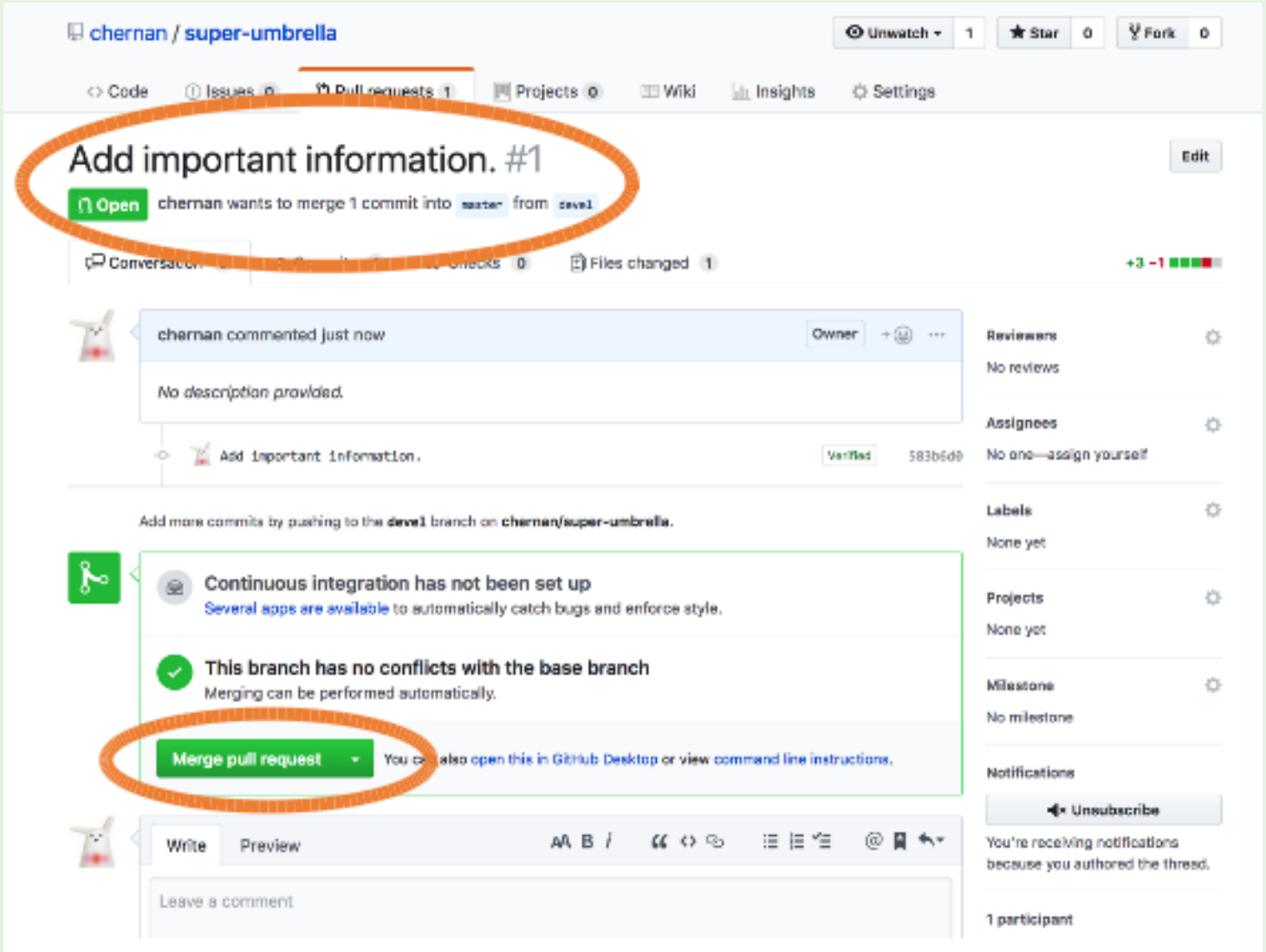
5: pull request, able to merge



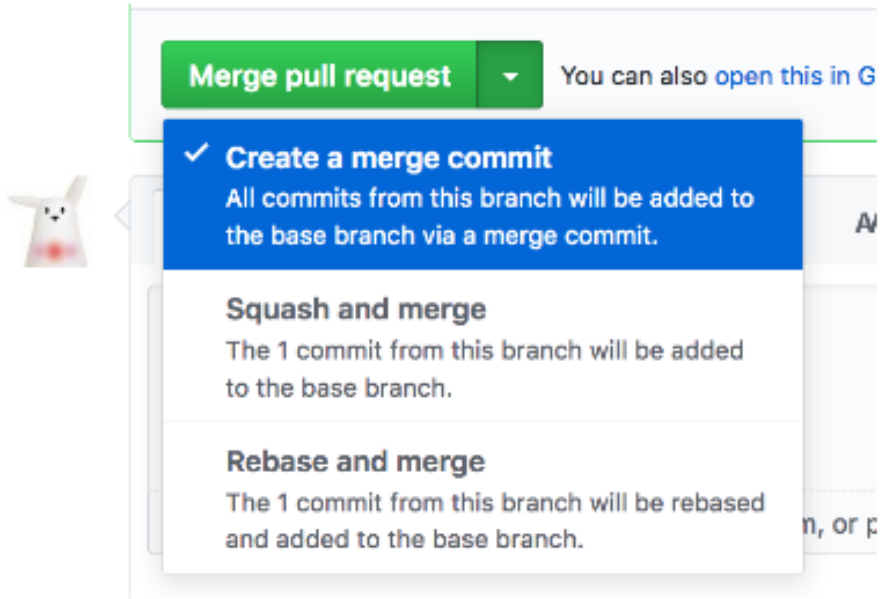
5: merge and pull request



5: merge and Pull request



5: merge and Pull request

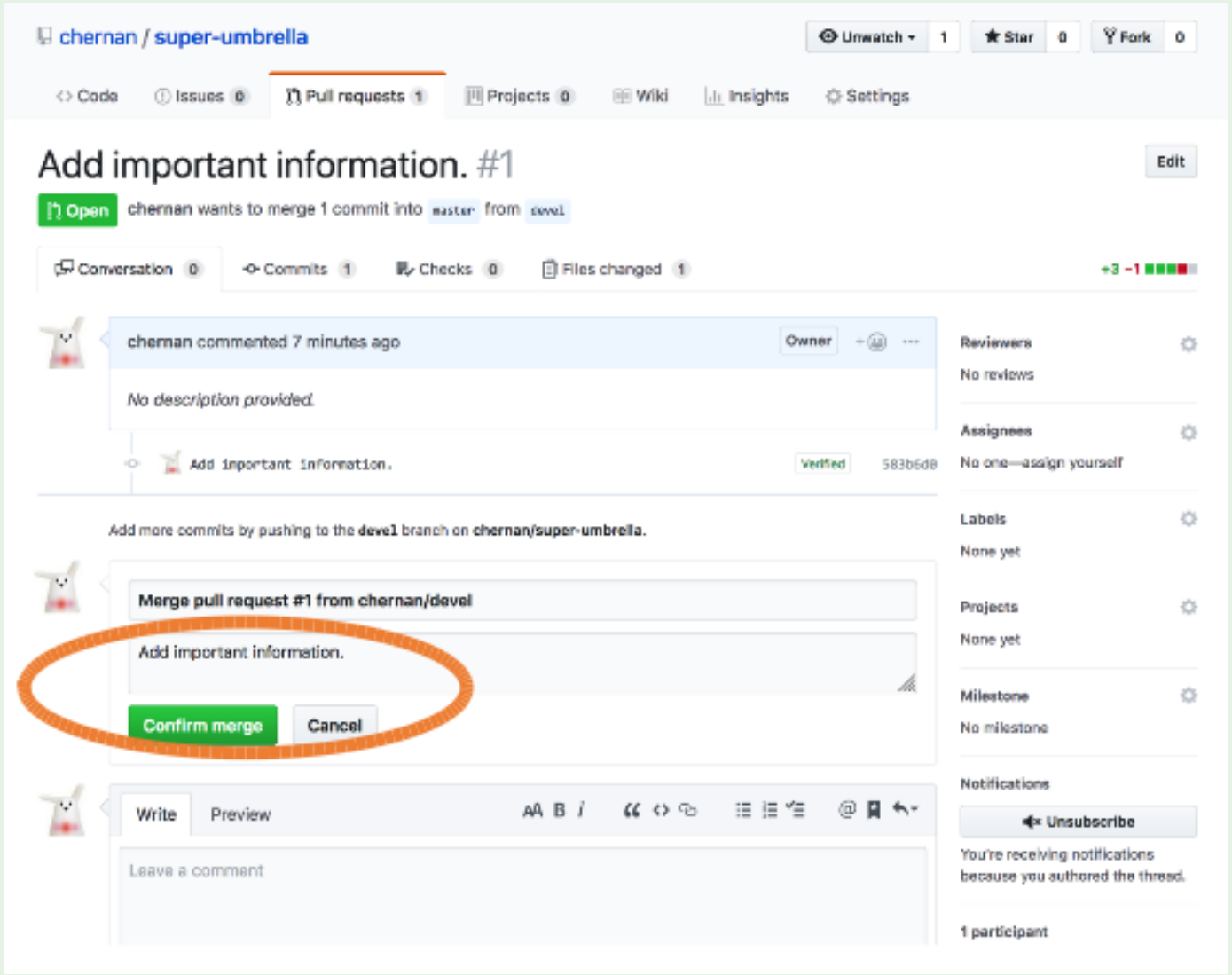


All individual commits are kept, branches are merged.

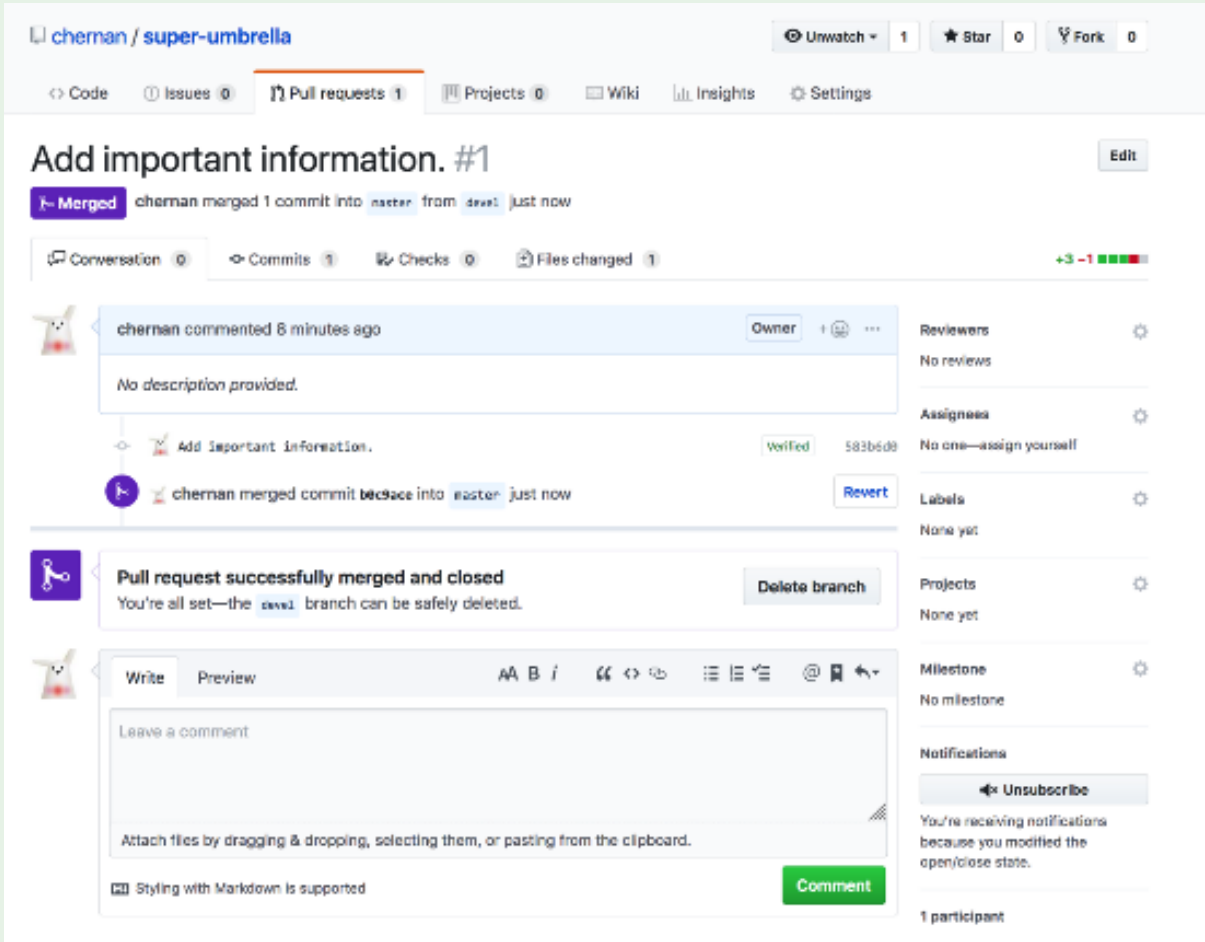
All commits are combined into one, and propagated to the base.

All commits are propagated to the base branch.

6: merge

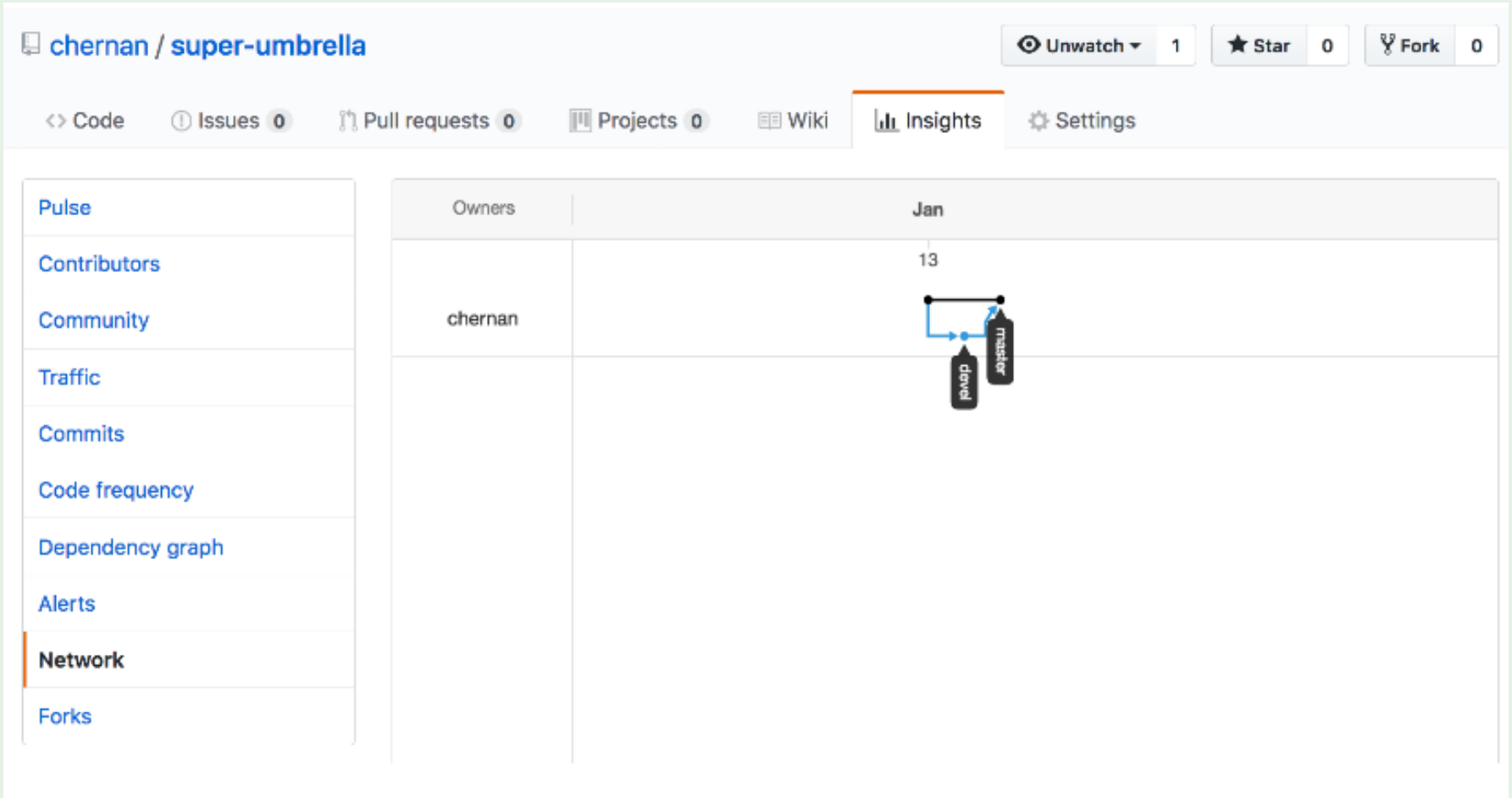

chernan / super-umbrella
Unwatch 1
Star 0
Fork 0
Code
Issues 0
Pull requests 1
Projects 0
Wiki
Insights
Settings
Add important information. #1
Edit
Open
chernan wants to merge 1 commit into master from devel
Conversation 0
Commits 1
Checks 0
Files changed 1
+3 −1
chernan commented 7 minutes ago
Owner
No description provided.
Add important information.
Verified
583b6d0
Add more commits by pushing to the devel branch on chernan/super-umbrella.
Merge pull request #1 from chernan/devel
Add important information.
Confirm merge
Cancel
Write
Preview
Leave a comment
Reviewers
No reviews
Assignees
No one—assign yourself
Labels
None yet
Projects
None yet
Milestone
No milestone
Notifications
Unsubscribe
You're receiving notifications because you authored the thread.
1 participant

7: the merge delete the branch



GitHub

See version network, Insights tab



GitHub Conclusion

GitHub GUI

- With this exercise, we modified a file in a directory of our own GitHub account.
- BUT: reserve this click button mode only for minor modifications (relies on a stable and smooth network connection!)
- Also, we collaborated only with ourselves
- In the next exercise, we will do this task again with a "git command line" mode and by collaborating all together.

GitHub Exercise 2

Previous exercise

In the previous exercise, we added change in the fork of our GitHub account through the GitHub GUI.

Objective

Now we will again modify a file but using a local working copy, so that we can work independently of the internet connection.
We will also collaborate all together (eg. the final README.md file should contains the name of all of us)
Steps of modifications will be done with git on a local clone while steps for collaborative building will be done through the GitHub interface.

Repository to fork:

We could use the previous forked repository to do the collaborative part, but as we want to practice changes in a local copy, we will fork another repository: clairetn, FAIR_bioinfo_github

Be added as collaborator to the repository

During the time you will made the change, I will invite you as collaborator in the Settings tab of this repository: need your github login.

GitHub

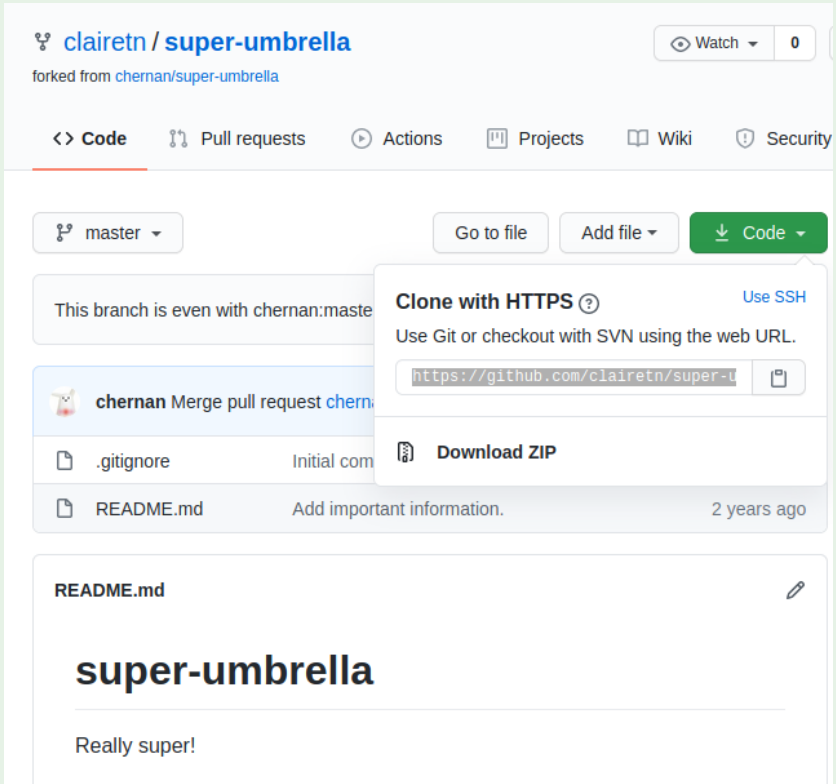
Steps

1. fork the repository on your github account (github fork)
2. clone your own forked repository in a new local working repository (git, local)
3. create a new branch (git, local)
4. do the modification (add your name in the README.md file) (local)
5. merge the branch (git local)
6. push the actual local version to our github repository (git local)
7. pull request the original github repository of our changing (github)
8. as a collaborator, push your changing in the original upstream repository (github)

GitHub: Cloning one of your fork

Github url:

- We clone a fork with the git command `git clone` followed by the url of the repository.
- This url is accessible with our mouse from the github repository (green "Code" button):



`https://github.com/clairetn/FAIR_bioinfo_github.git`

GitHub: Git commands

GUI Github → local git (just done):

```
git clone <url_of_your_github_account>
```

Local work:

```
git branch mybranch
git checkout mybranch
# do change, eg. add your name in README.md file
git add README.md
git commit -m "add name"
git checkout master
git merge mybranch
git delete mybranch
```

Local git → GUI Github:

```
git push origin master
```

GitHub: Propose your change into the initial project

From your forked repository

- "Compare" and then "Pull request" your issue (explain your proposals as much as possible)
- conflicts when one change the same line
- manage possible conflicts with the Github GUI

Many small commits

⇒ do many small commits easier to merge than a big unique one

Bonus

Challenge

- make a (voluntary today) "error" by suppressing the new dedicated repository created for this git exercise
- retrieve your code with the git clone command on your github repository

Conclusion

Git points

- no possibility to make merge when the file are not in text format
- ignore some files from tracking: create a .gitignore file containing one file/repository name by line (wildcards accepted)
- tagging a commit: fix a version as a reference
- editors with integrated git: Atom, Visual Studio Code, ...

Github points

- github offers more than git embedded commands: sharing code with others, web pages, continuous integration, and more
- pull requests is the way to collaborate

Ressources

- Learning Git by Software Carpentry: `https://swcarpentry.github.io/git-novice/`
- Git Cheat Sheets: `https://services.github.com/on-demand/resources/cheatsheets/`
- A step-by-step progression to link RStudio and GitHub: `https://jules32.github.io/2016-07-12-Oxford/git/`
- Pierre Poulain fr ressources: `https://cupnet.net/git-github/`