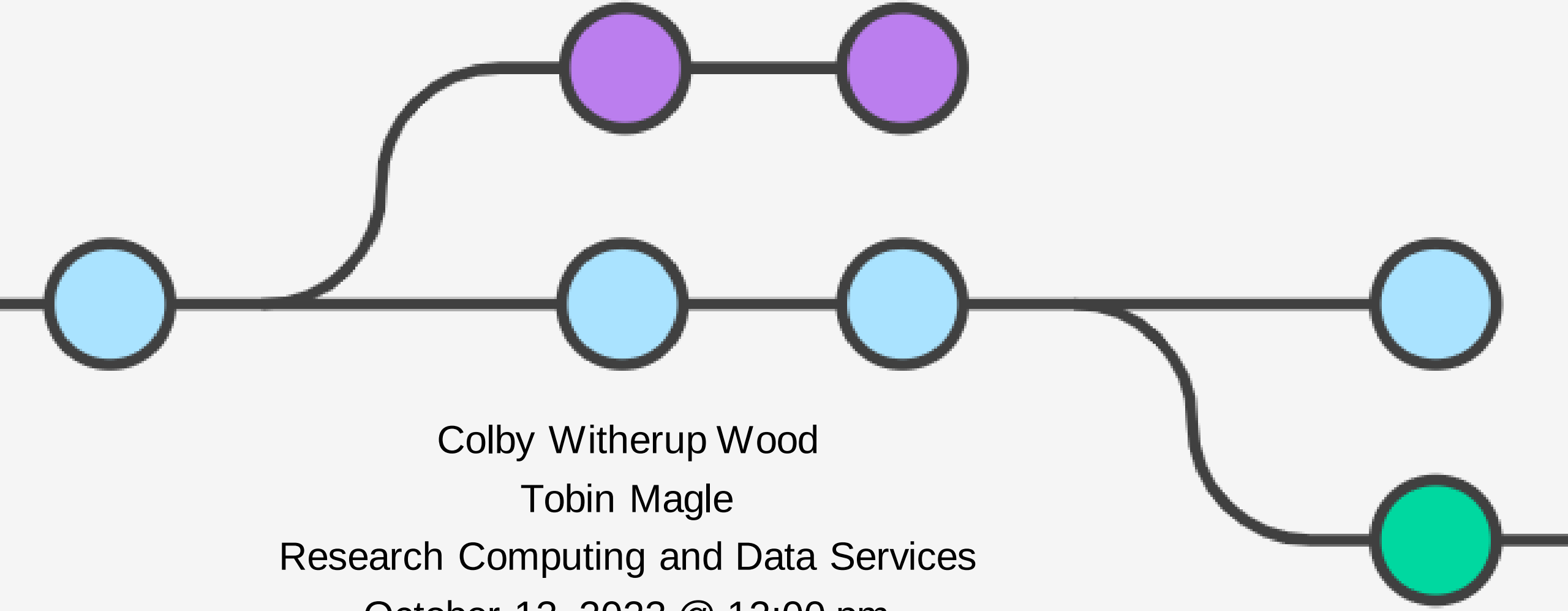
Workshop materials

- https://github.com/maglet/github-gui-demo

Version control using GitHub



Colby Witherup Wood

Tobin Magle

Research Computing and Data Services

October 13, 2023 @ 12:00 pm

Today we're going to cover:

- What is **version control**?
- What is **git**?
- Version control using **GitHub**
- **Collaboration** in GitHub

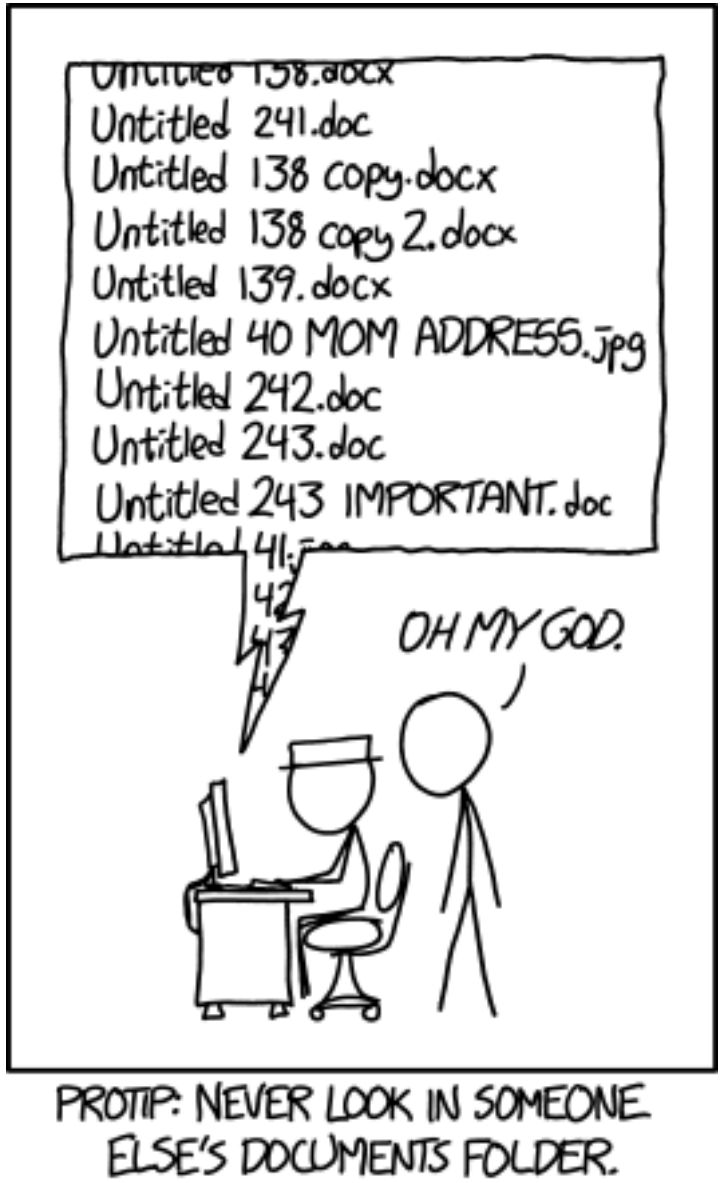
How to get help after today's talk

- bit.ly/rcdsconsult
- We provide free consultations for programming, data science, and data management issues that are related to research

Manual Version Control

“Renaming files”

- Creates many files
- Relies on your ability to name files consistently
- Not designed for collaboration



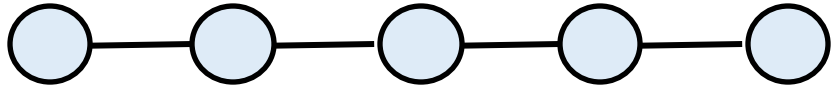
https://xkcd.com/1459/

Version Control Systems

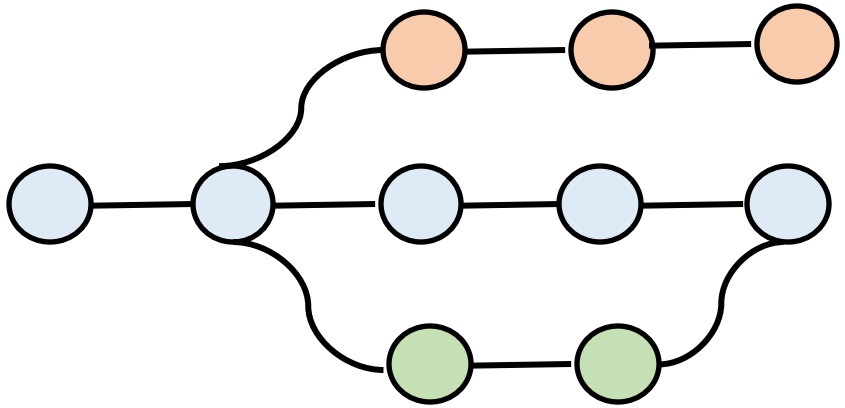
“Unlimited undo”

- One file, many versions
- Records who made what change when
- Good for **collaboration**: identifies and helps resolve conflicting changes

Solo

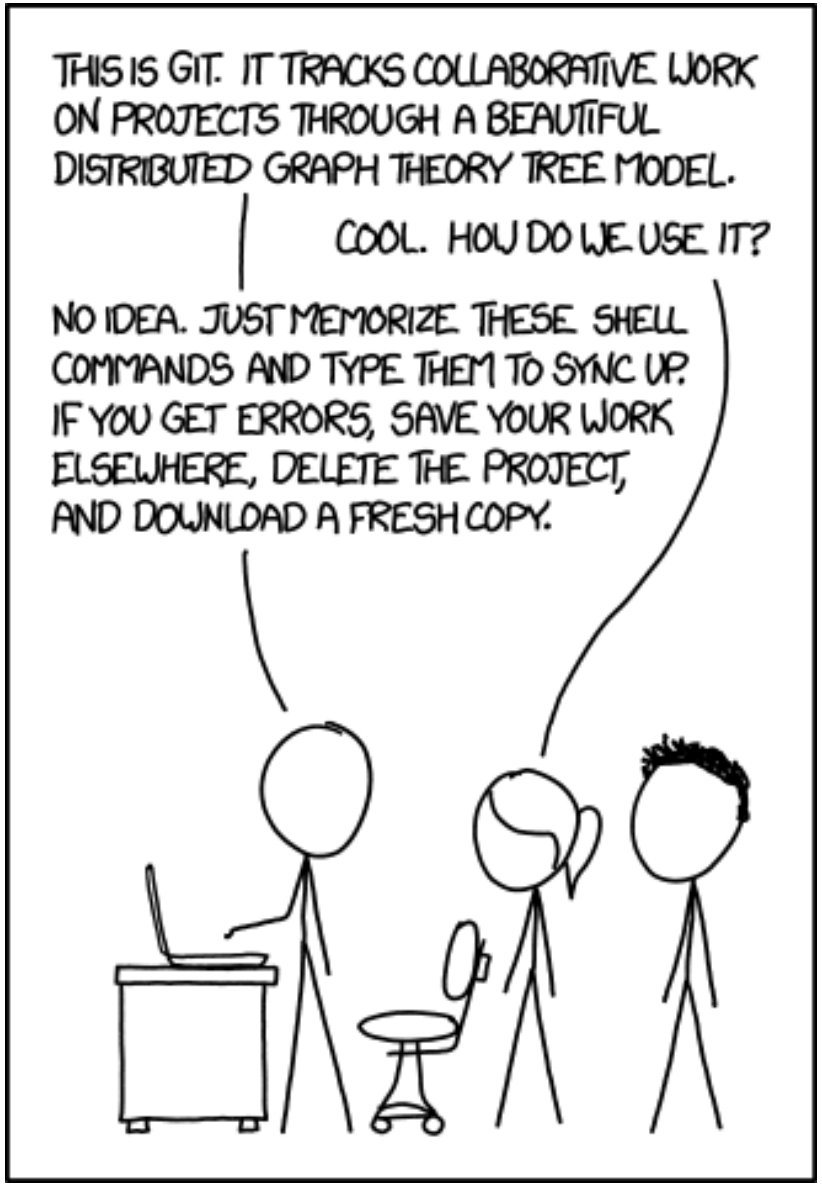


Collaborative



Today we're using **git**

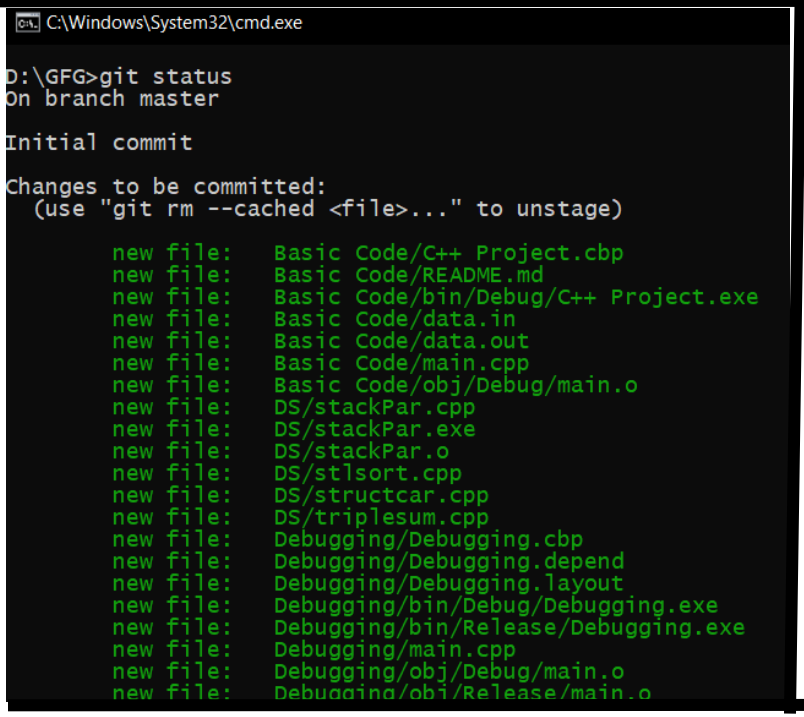
- Built for large software development projects
- Can get complicated, but the basics are simple
- Free, open source
- Many online resources/tutorials



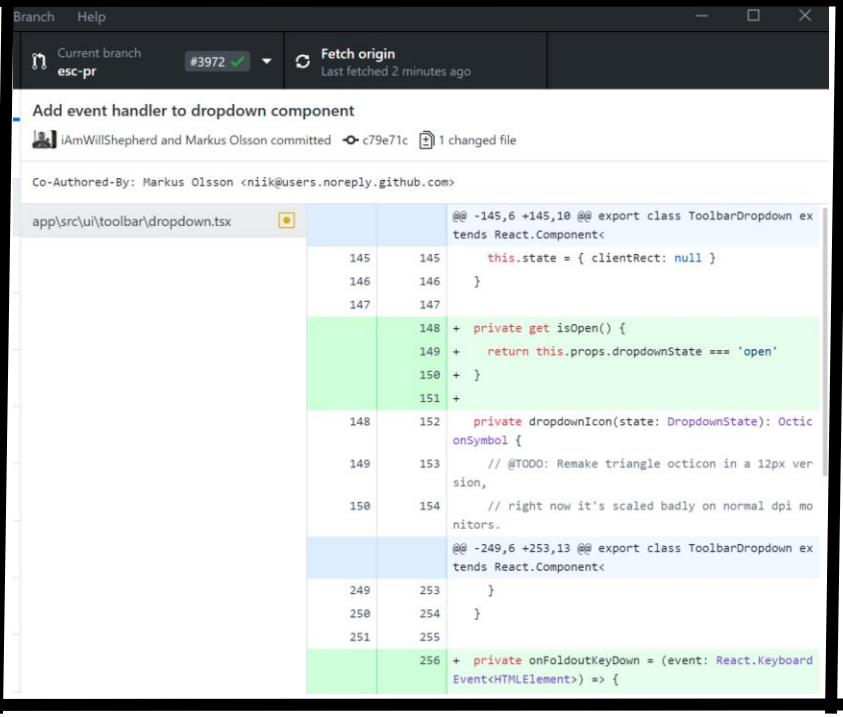
https://xkcd.com/1597/

Ways to use Git

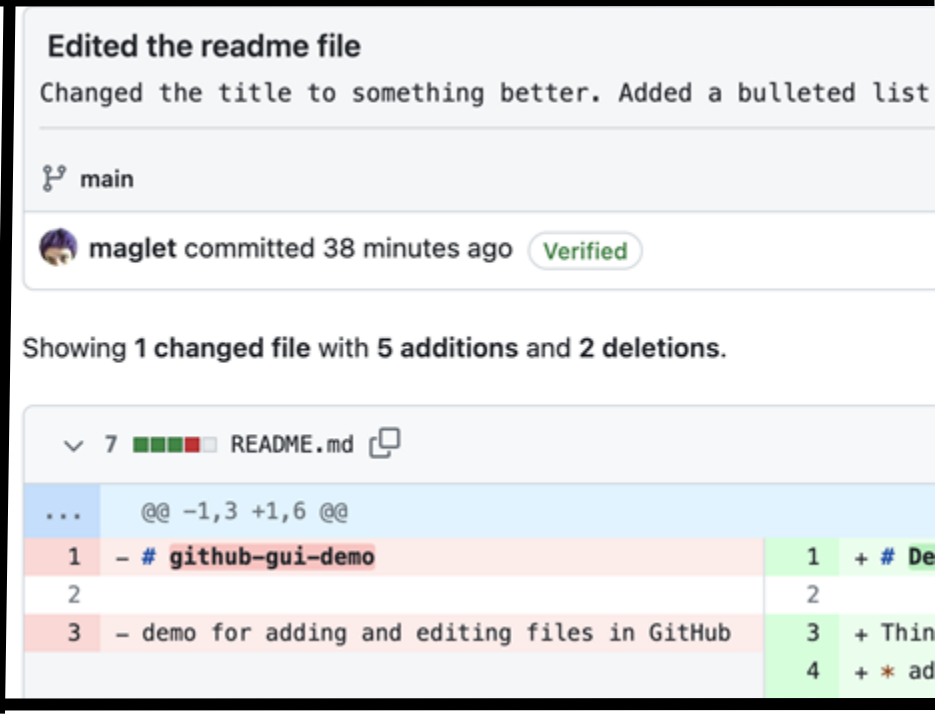
Command line



Desktop Clients

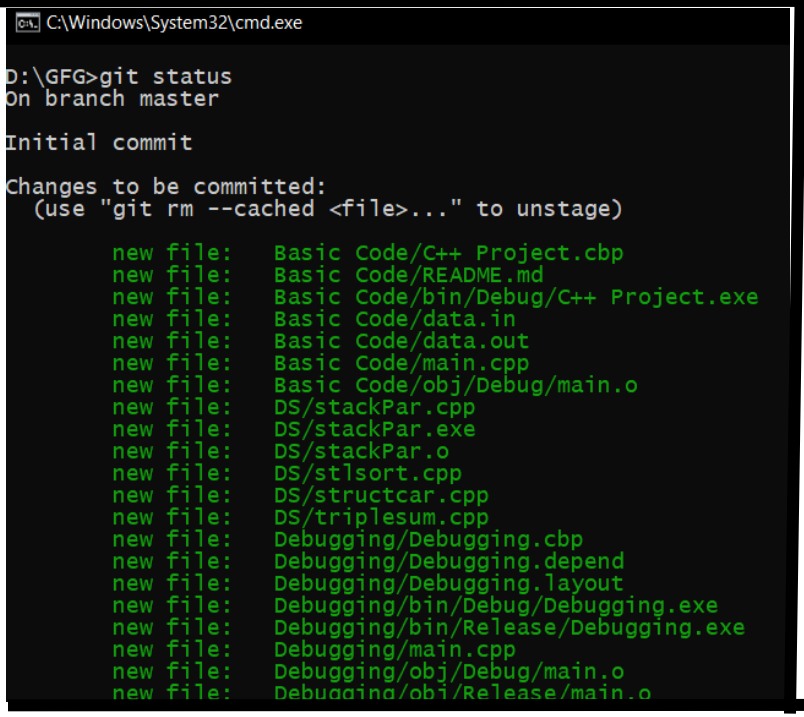


GitHub Website

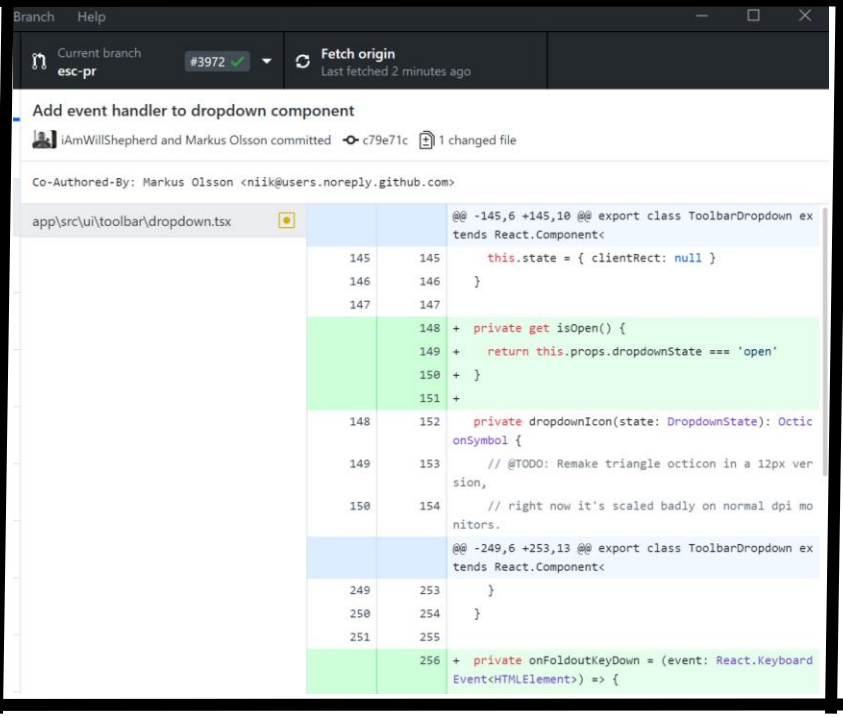


Ways to use Git

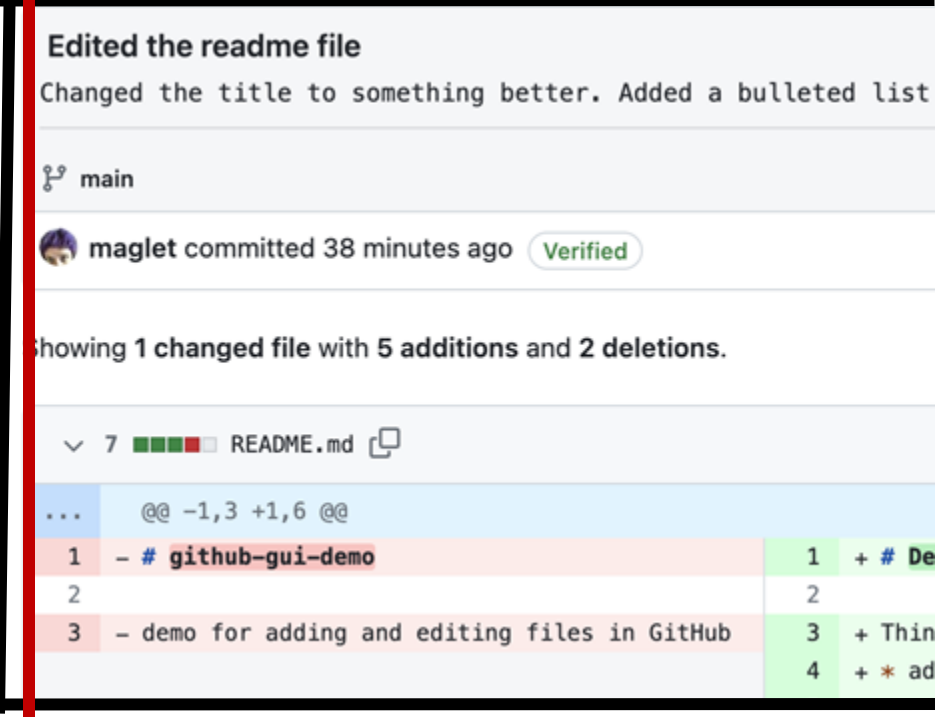
Command line



Desktop Clients



GitHub Website



GitHub is a place to

- Store your work remotely
  - Free public repositories
  - Unlimited private repositories with Northwestern GitHub Org
- Edit files in the web browser
- Collaborate

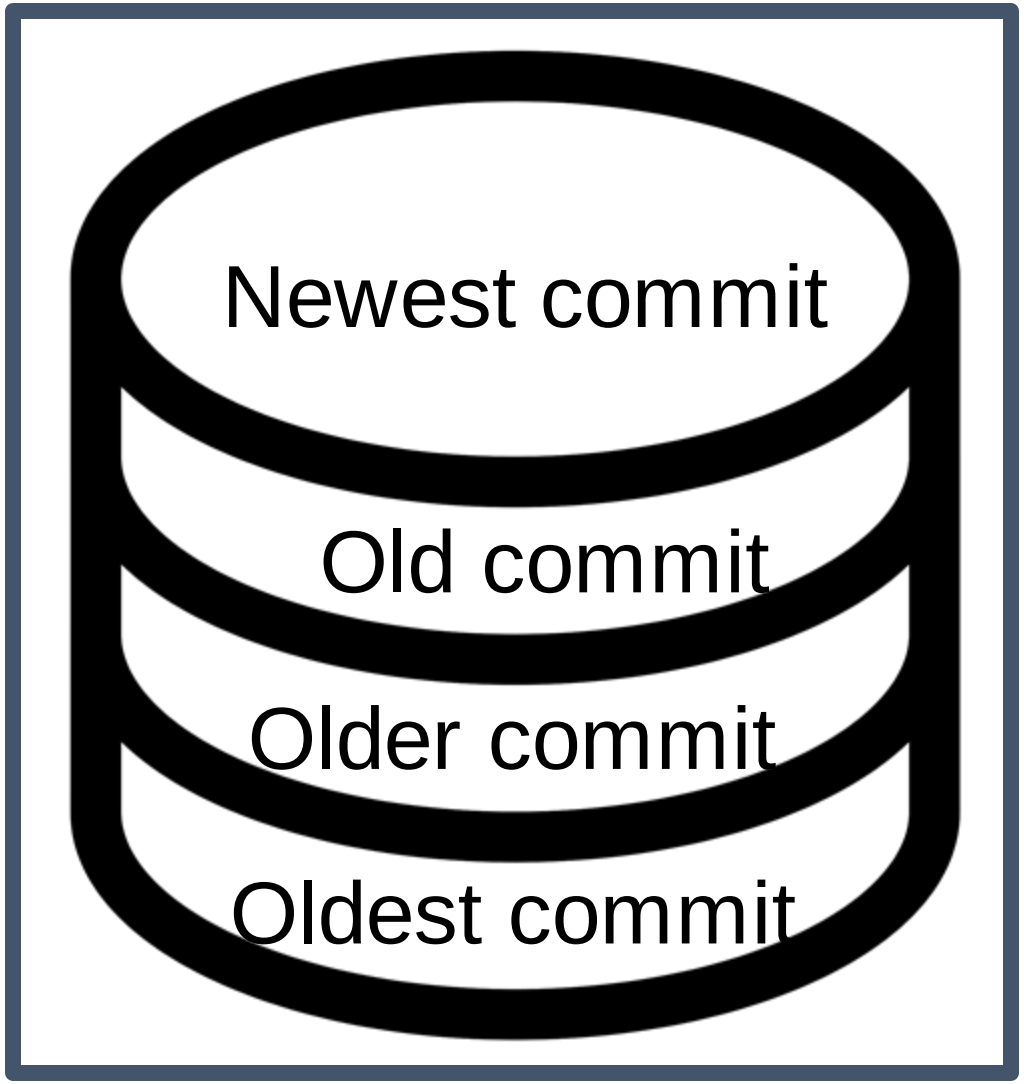
https://services.northwestern.edu/TDClient/30/Portal/Requests/ServiceDet?ID=55

GitHub best practices

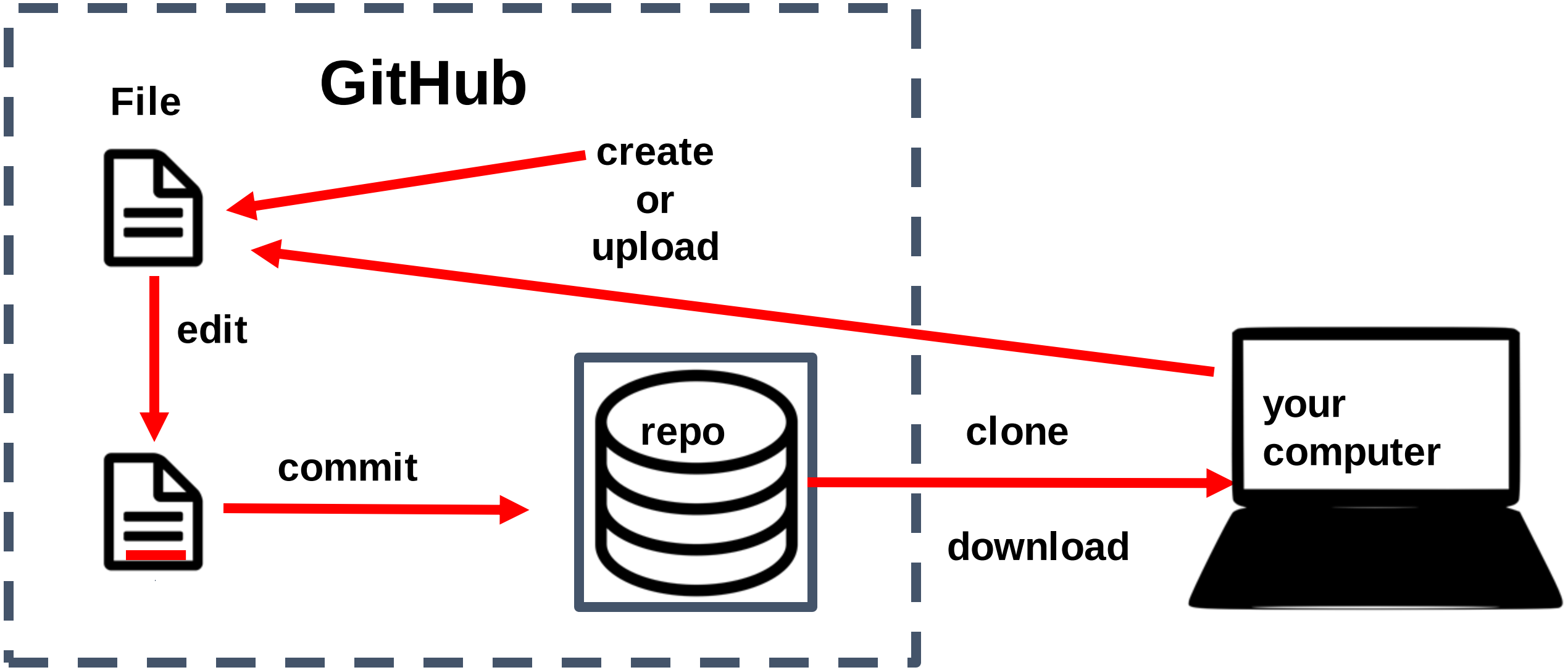
- GitHub is for code and documentation, not data
- Never put protected or personal data on GitHub, even in a private repo
- Store credentials (private keys or tokens) in a separate file outside of your repository (ie, not in your code).
- Add a personal email to your GitHub account, so you can use it when you leave Northwestern

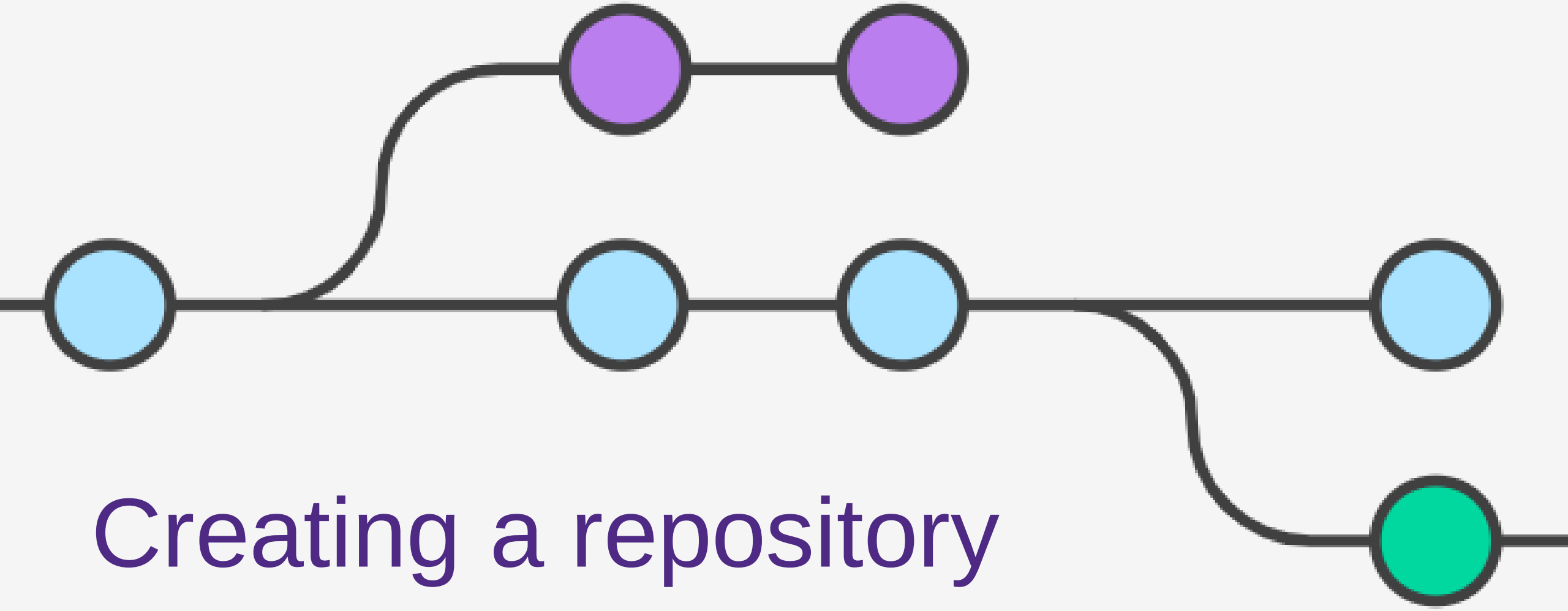
GitHub jargon

- **Commit** - save your changes to the files in the repository – each has a unique id
- **Repository** (“repo”) - a running log of changes that were committed - all you see is the most recent version
- **Blame** – who changed what
- **History** - a list of commits
- **Diff** - compare what has changed between versions line by line



GitHub workflow overview

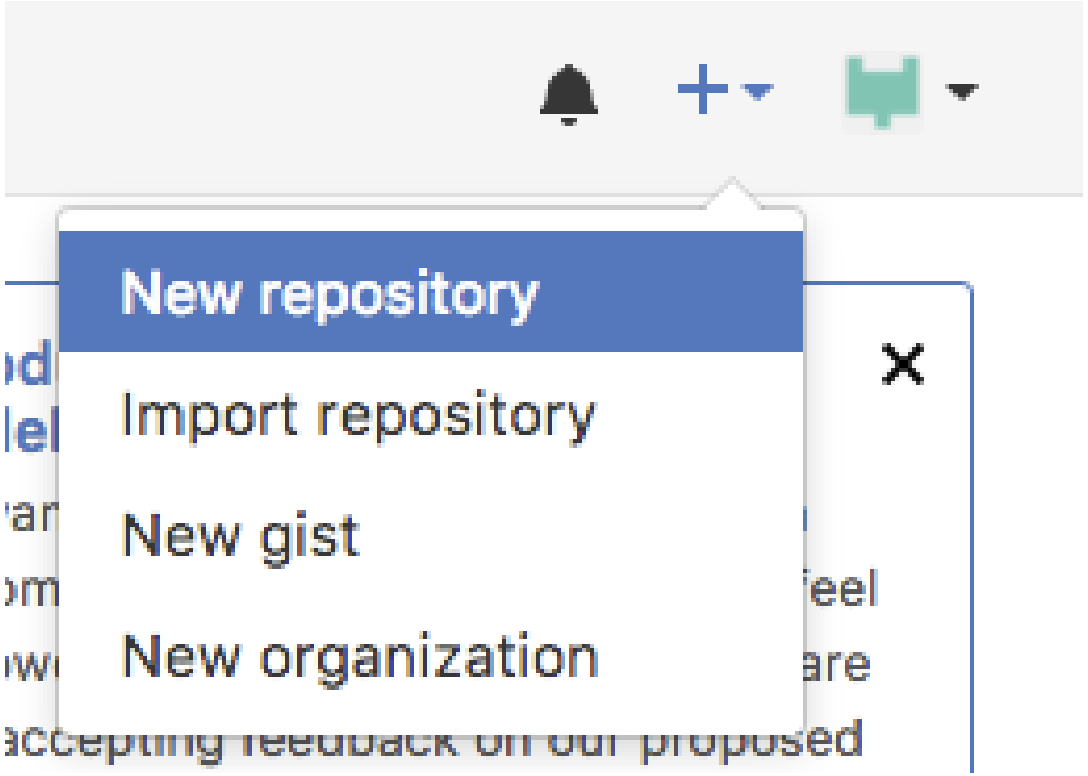




Creating a repository

Make an empty GitHub repo

- Go to http://github.com
- Click the plus sign on the upper right hand side of the screen
- Select New repository



Make an empty repository

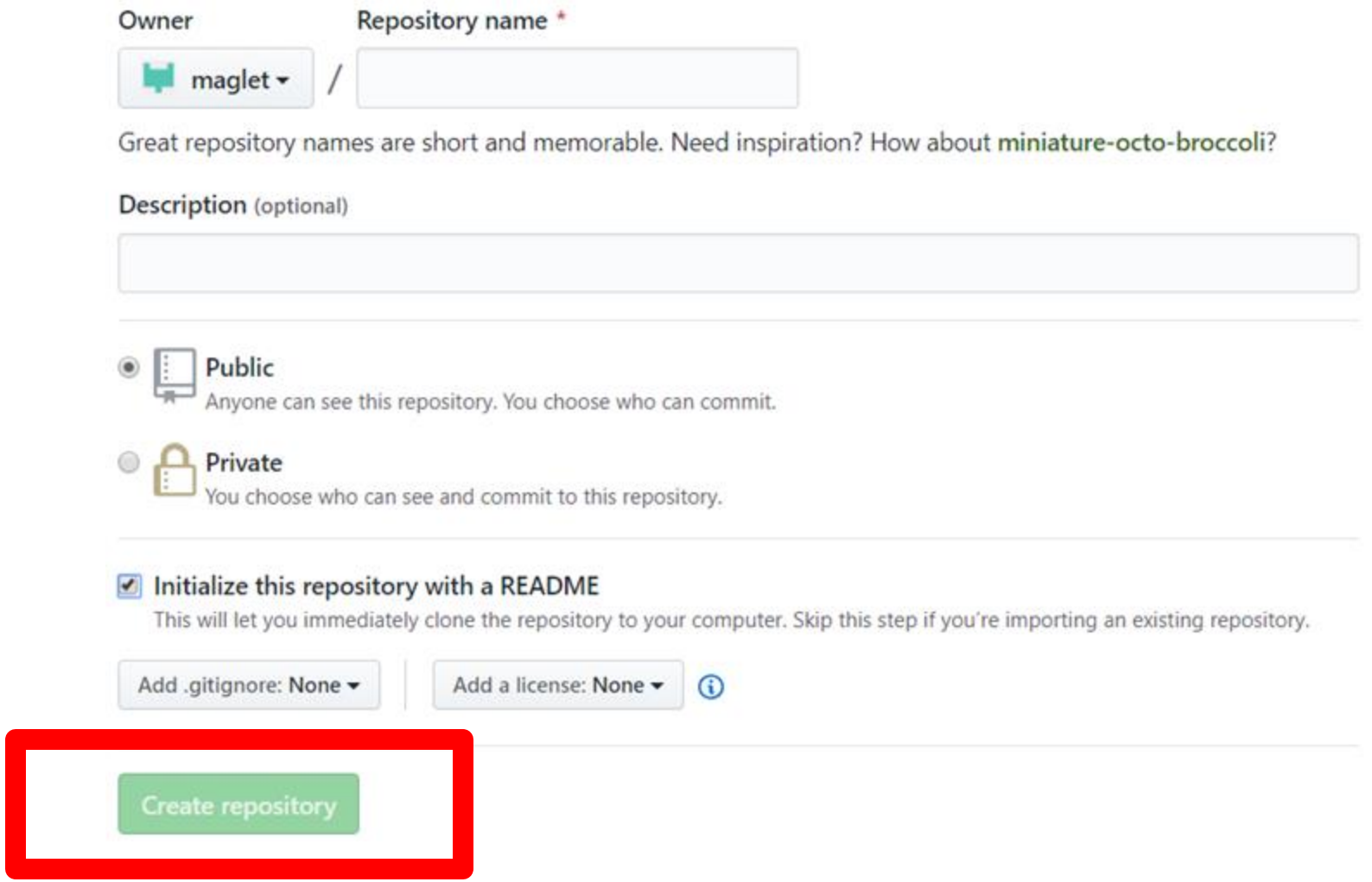
Name your repository

Provide a description

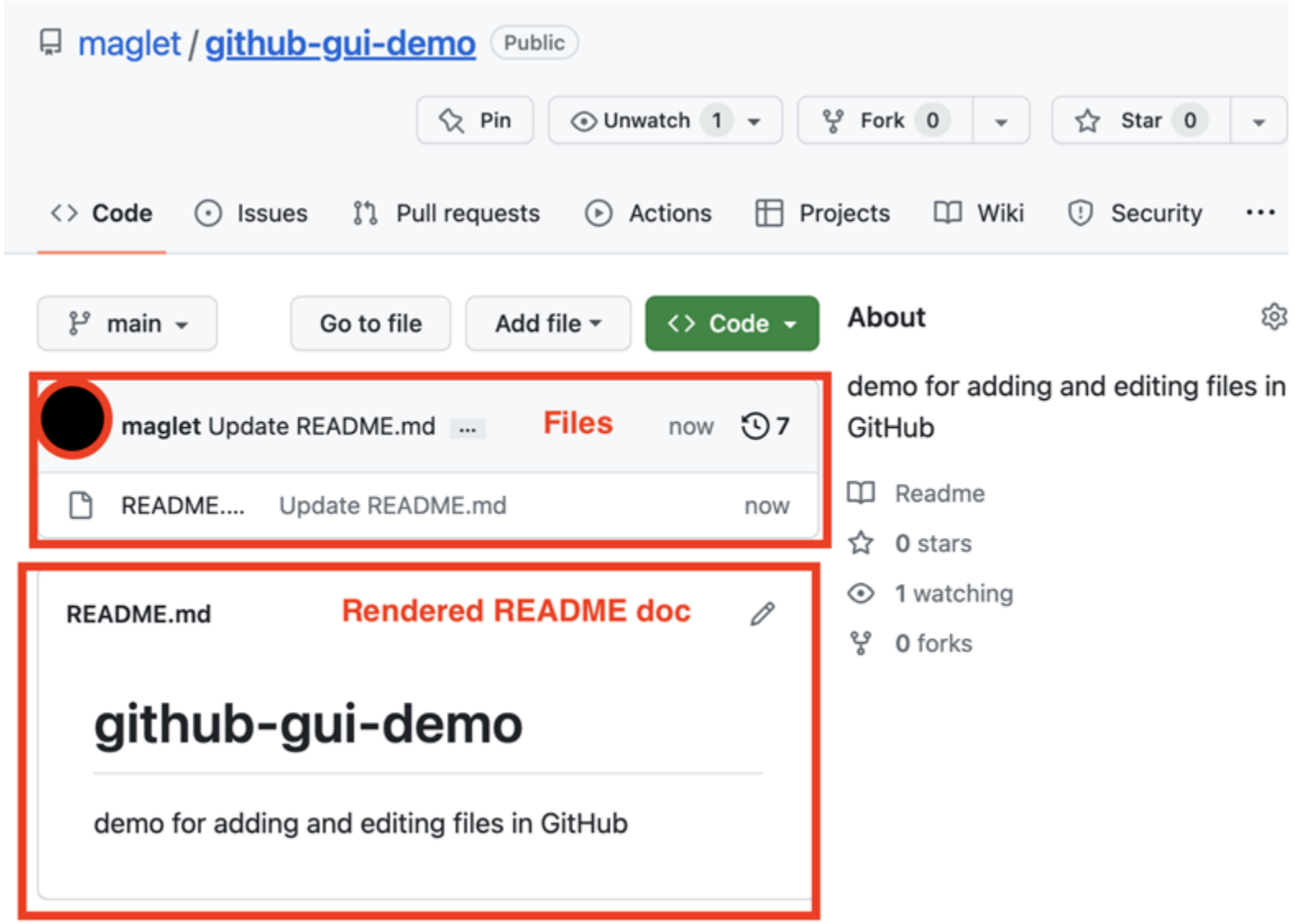
Keep it public

Initialize with a README

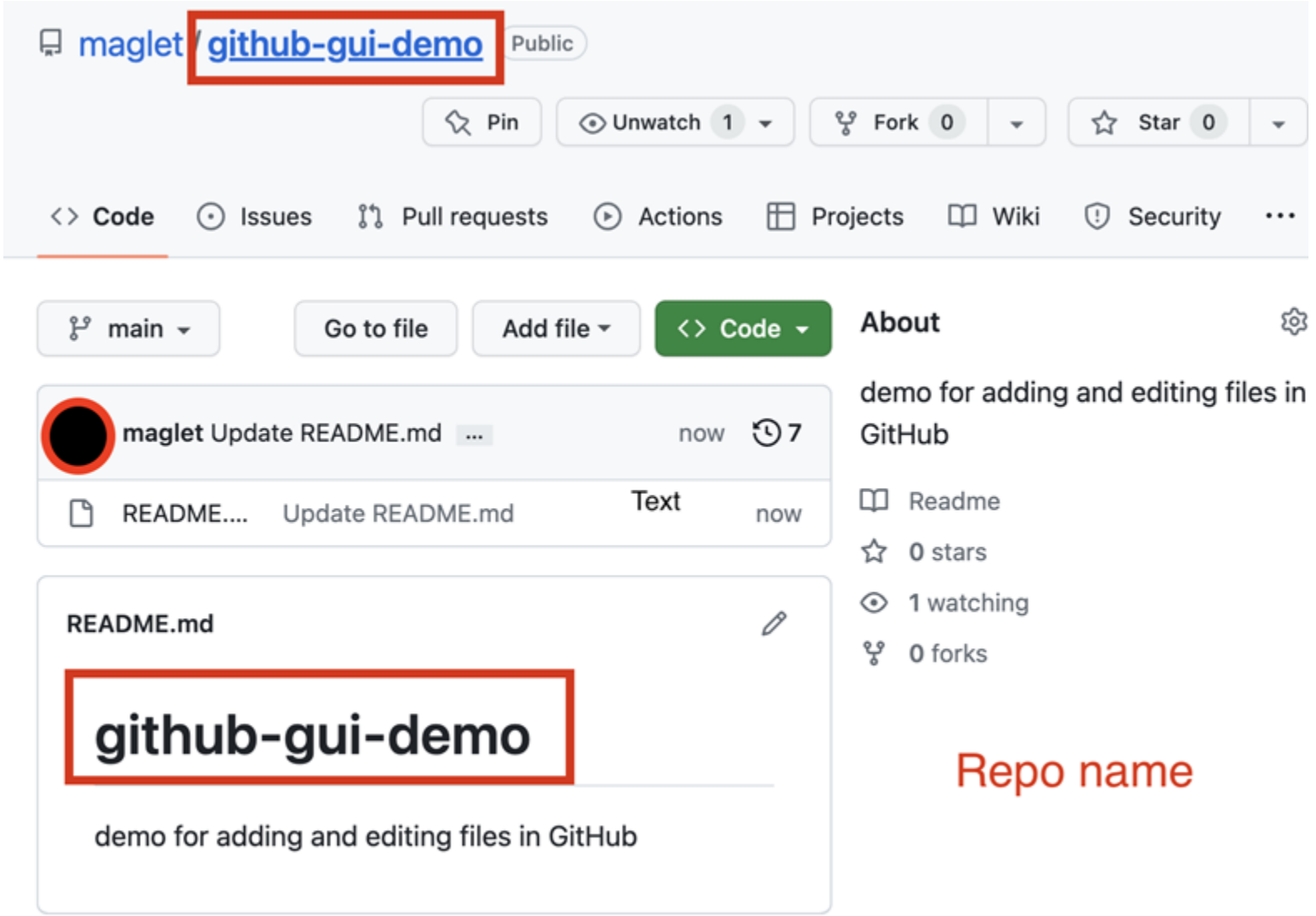
Create a repo



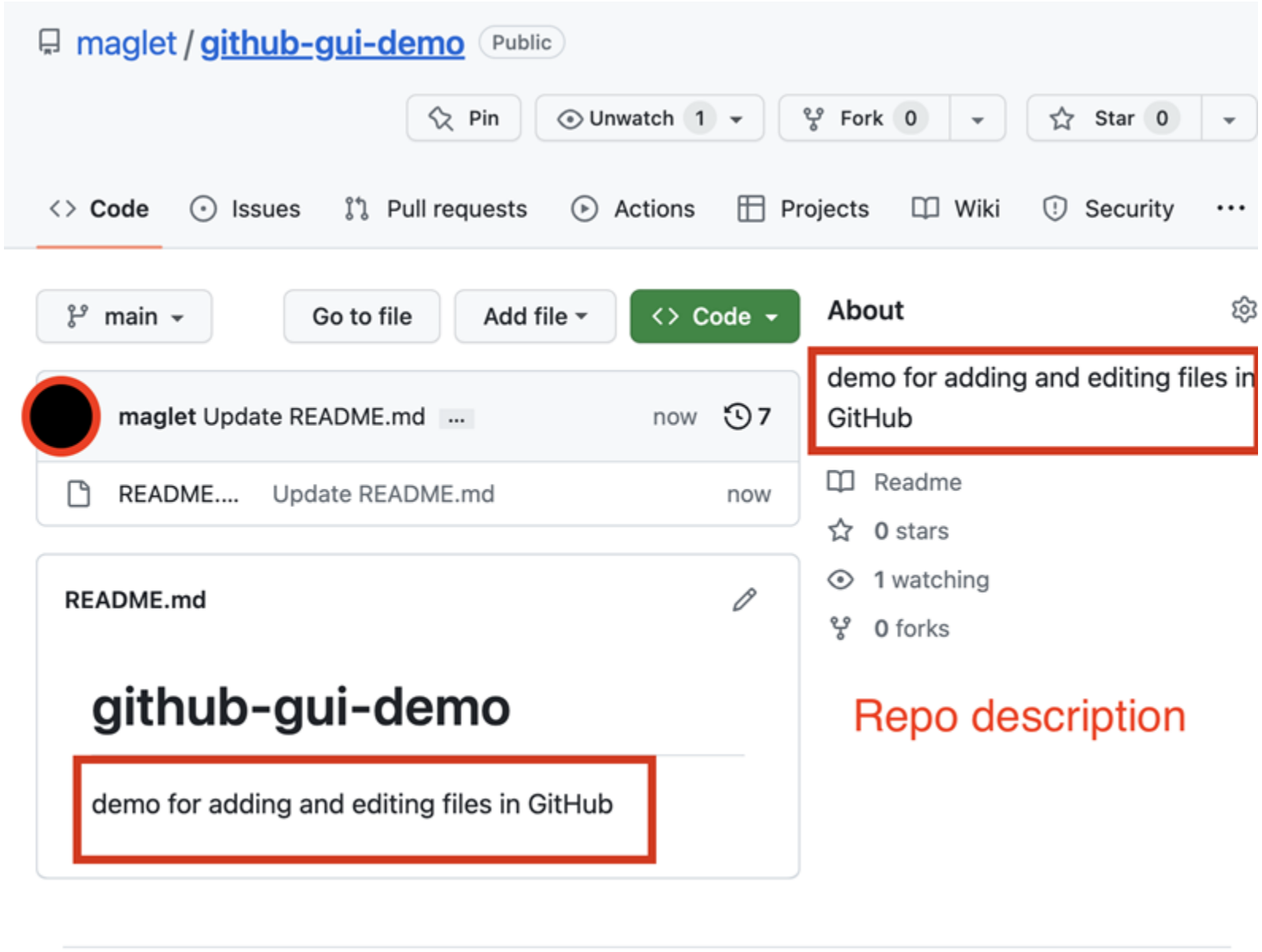
What you should see



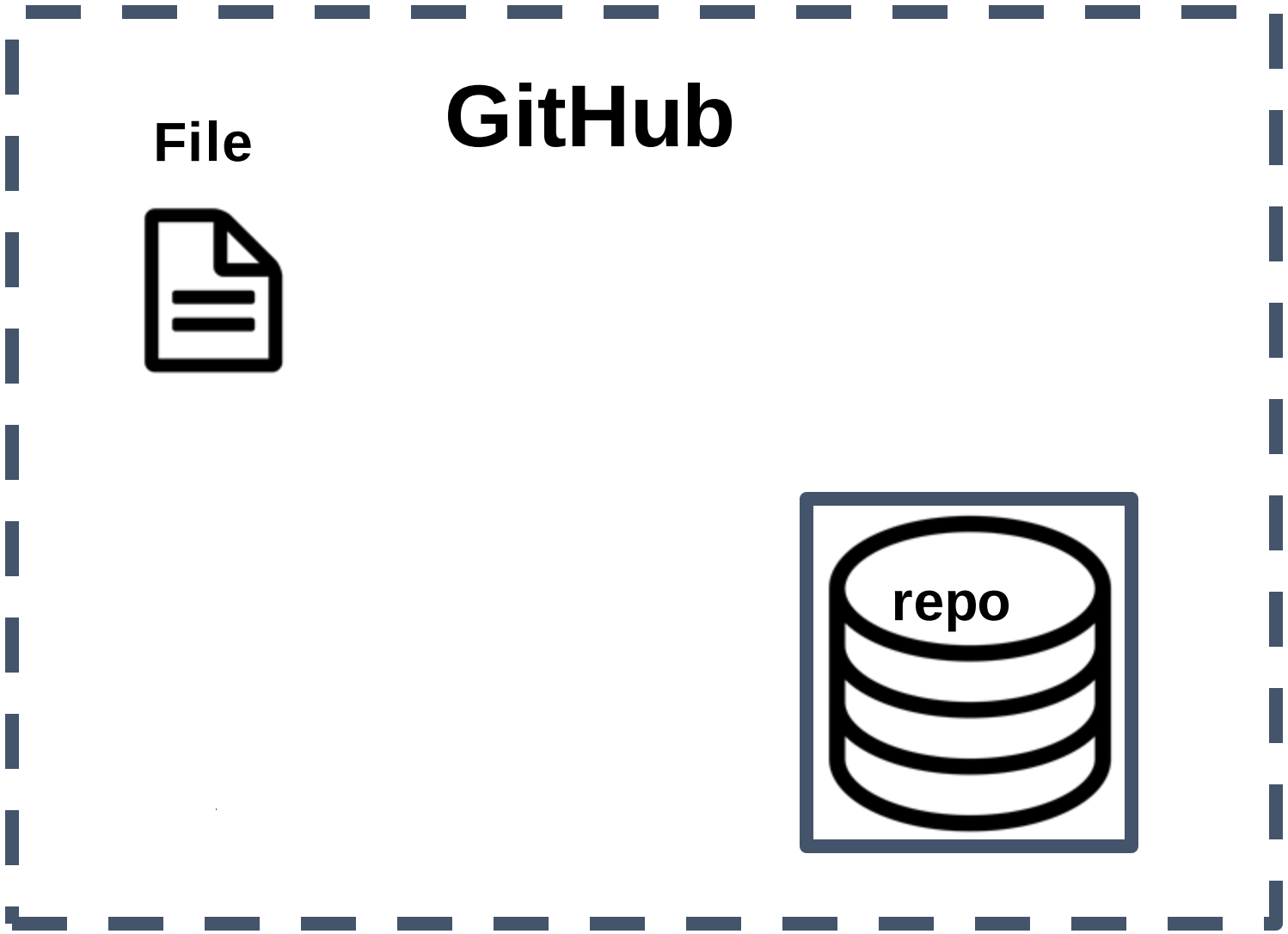
What you should see

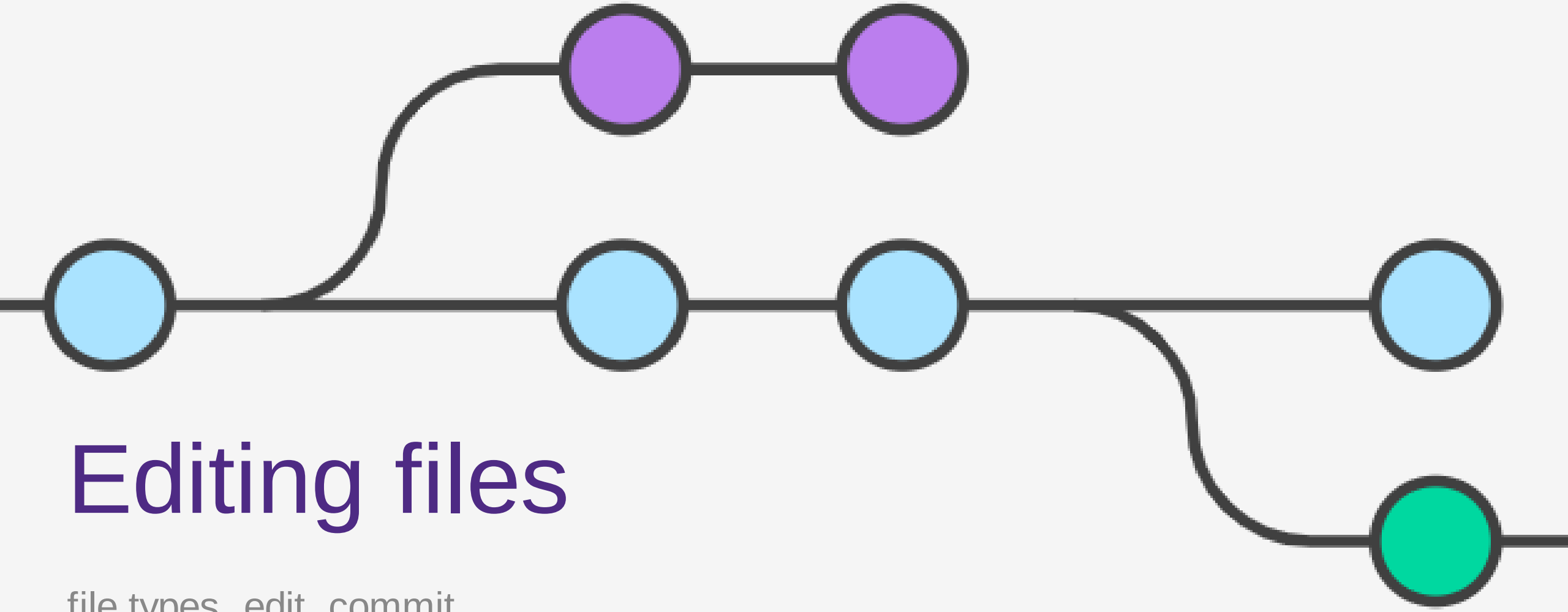


What you should see



What we've done so far





Editing files

file types, edit, commit

Types of files git works best with

“Good” formats (text based)

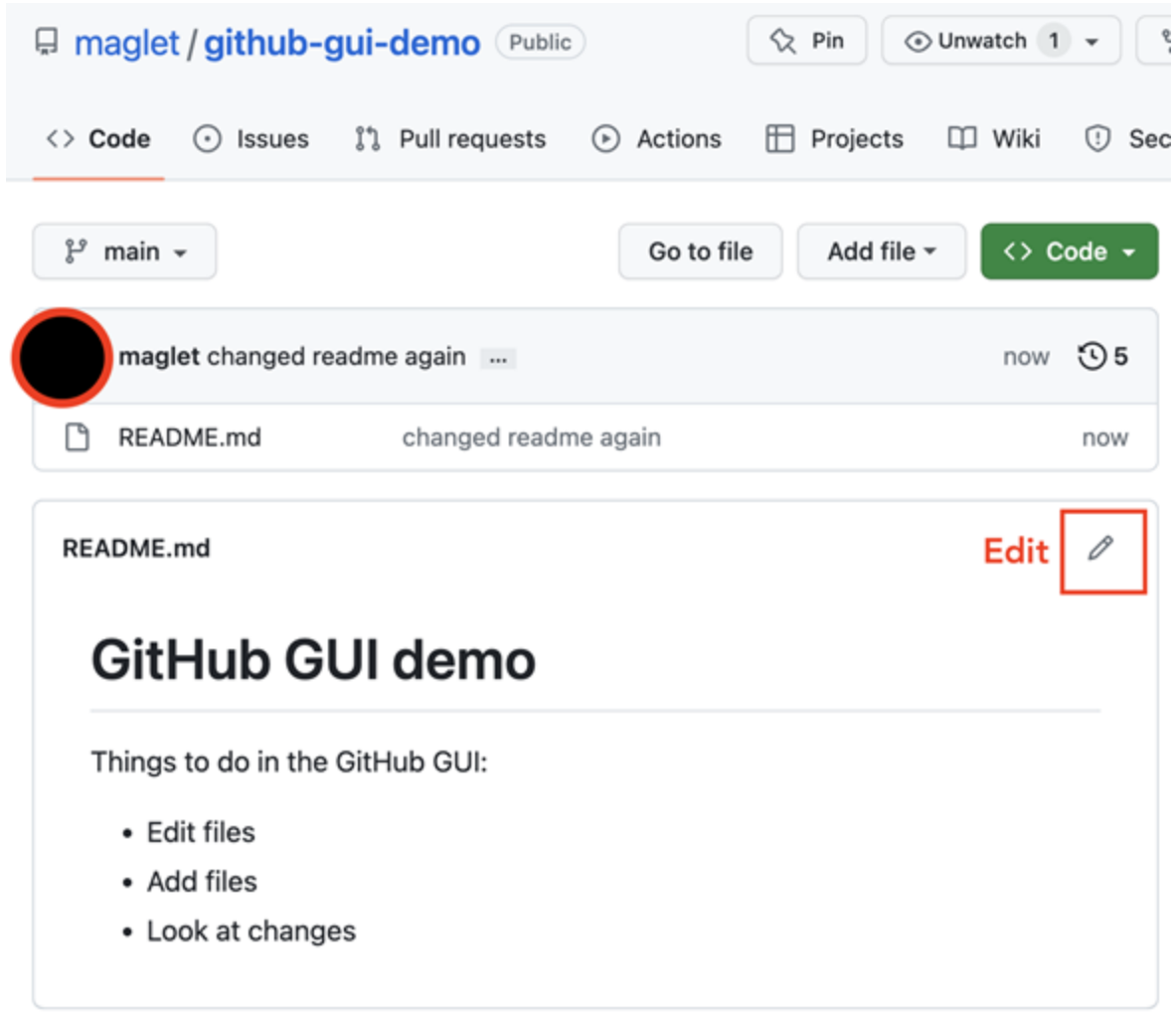
- Documents (.txt, .tex, .rtf, .md)
- Tabular Data (.csv)
- Source code (.R, .py, .c, .sh)

“Bad” formats (binary files)

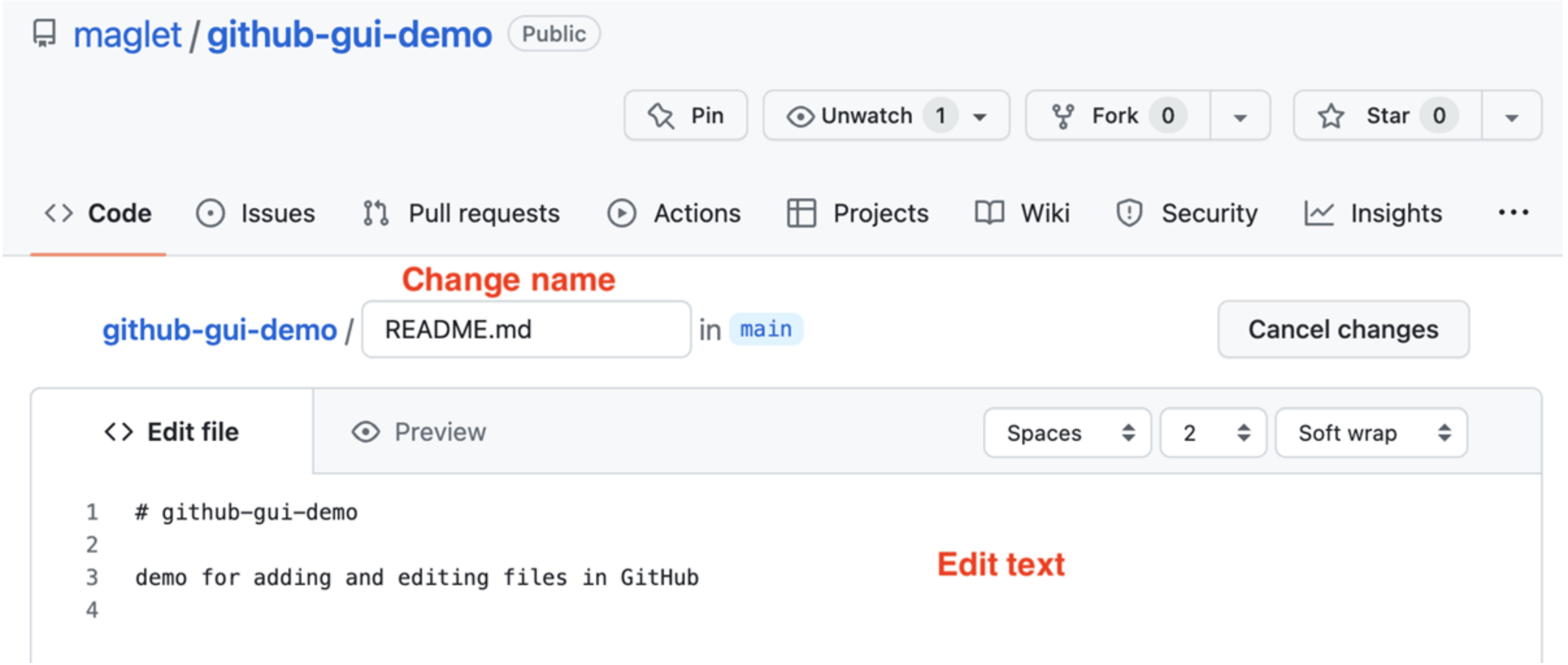
- Documents (.docx, .pdf, .ppt)
- Excel spreadsheets (.xlsx)
- Media (.jpg, .mp3, .mp4)
- Databases (.mdb, .sqlite)

https://www.nayuki.io/page/what-are-binary-and-text-files

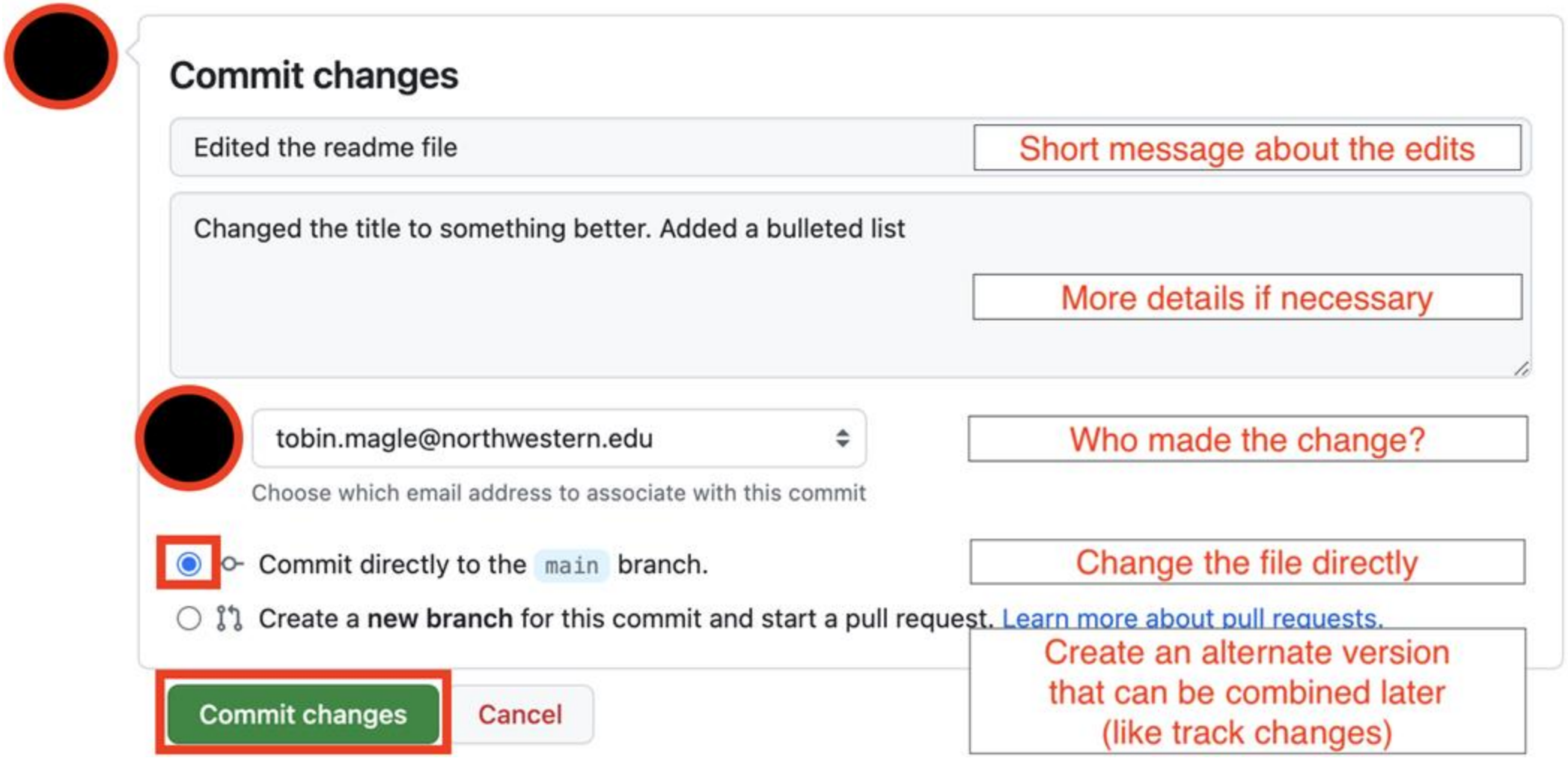
Edit the README file



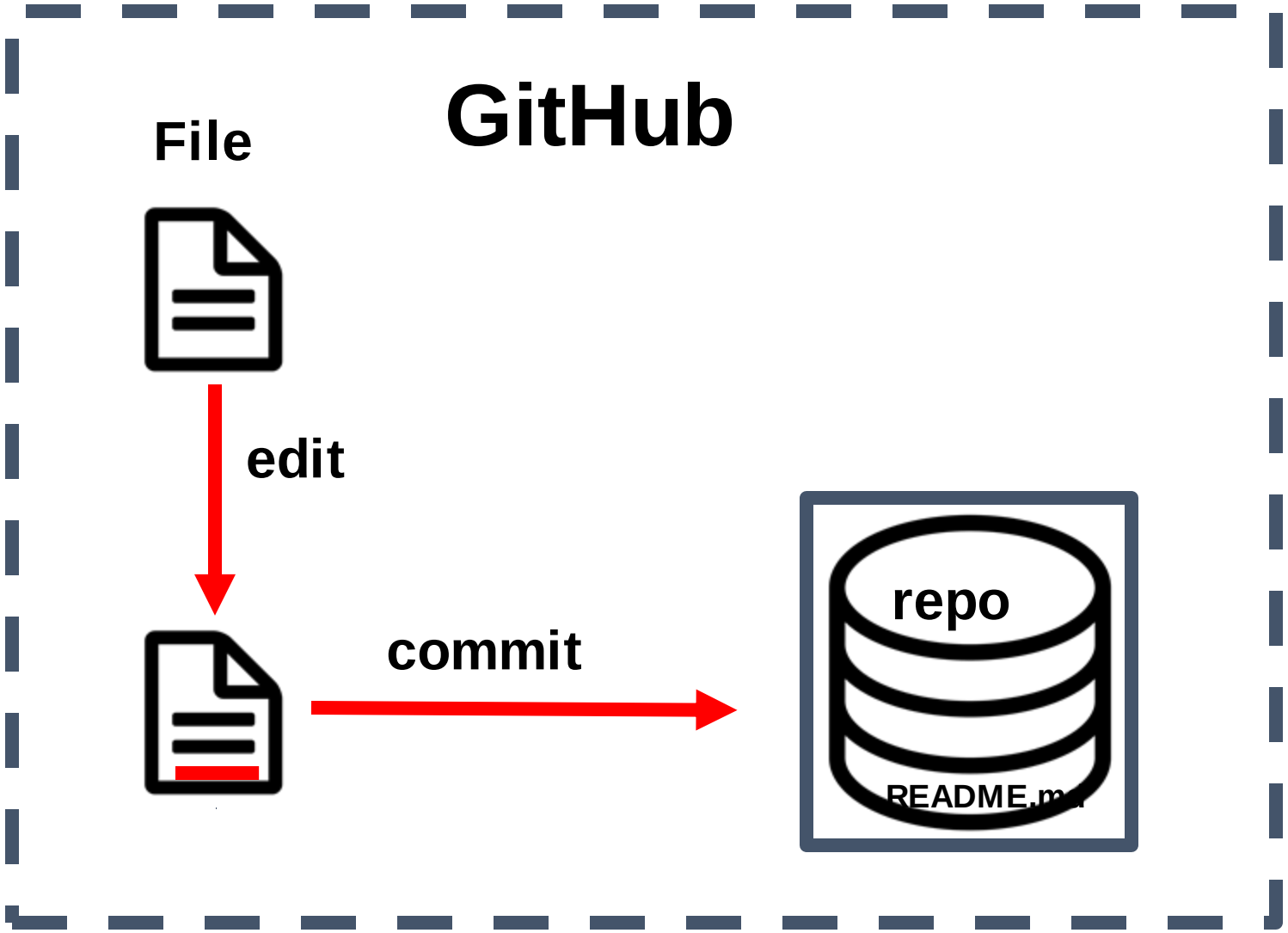
File editor

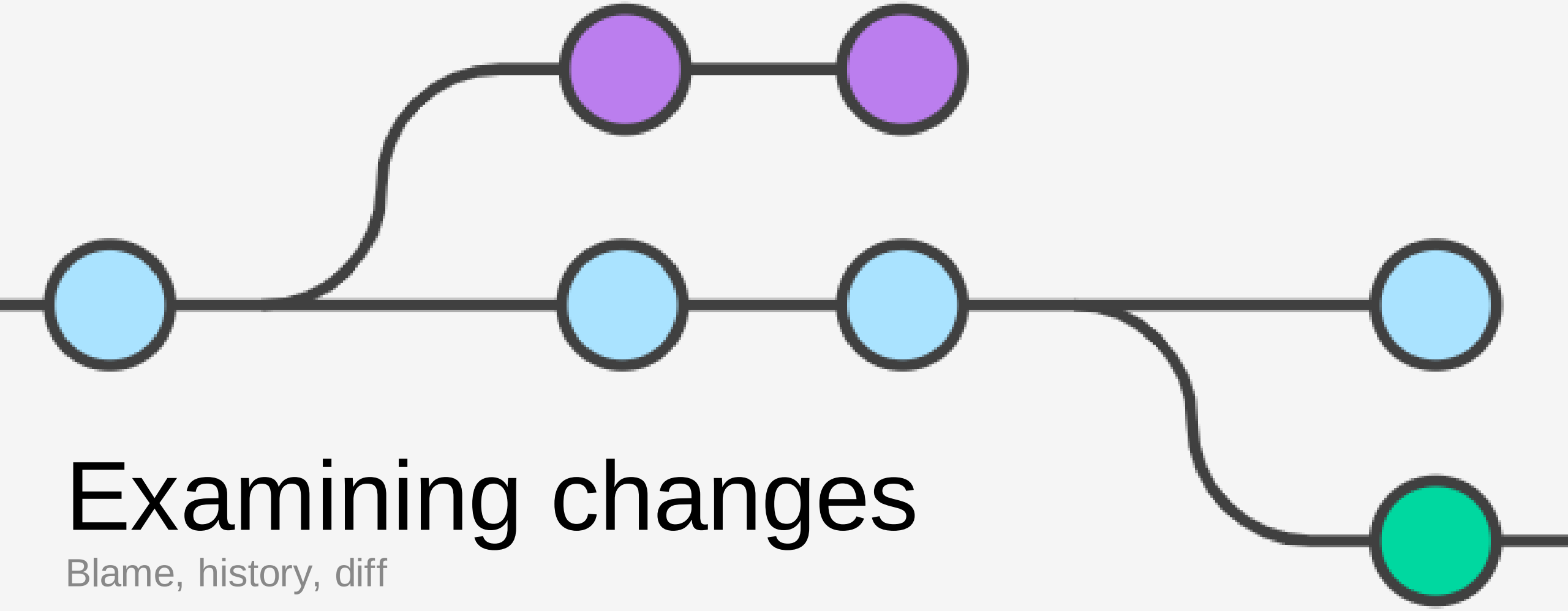

maglet / github-gui-demo
Public
Pin
Unwatch 1
Fork 0
Star 0
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Change name
github-gui-demo /
README.md
in
main
Cancel changes
Edit file
Preview
Spaces
2
Soft wrap
1 # github-gui-demo
2
3 demo for adding and editing files in GitHub
4
Edit text

Commit Changes



What we've done so far



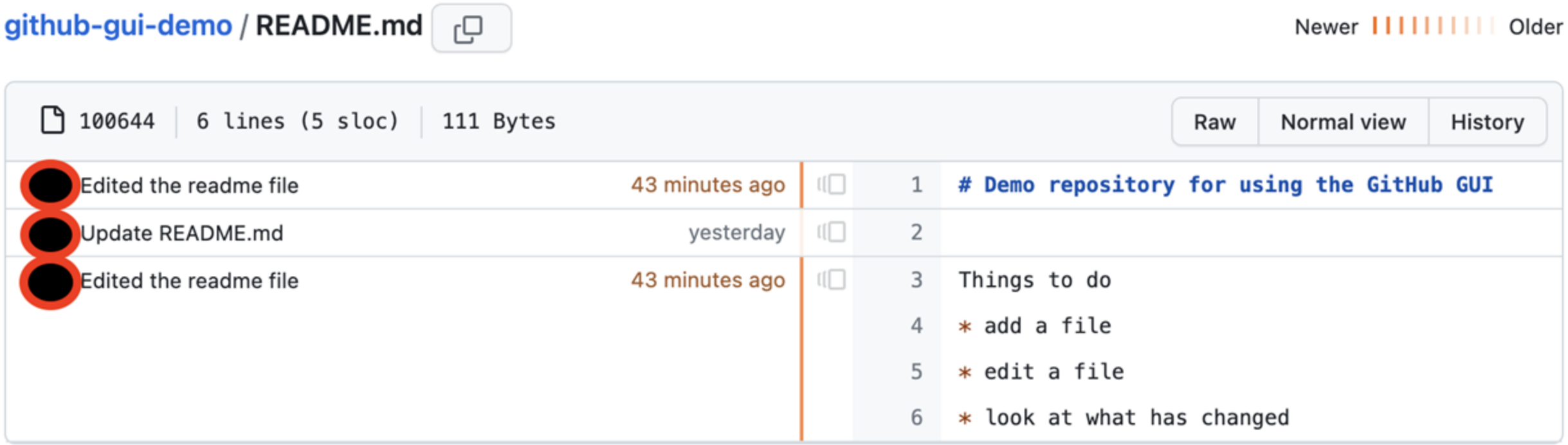


Examining changes

Blame, history, diff

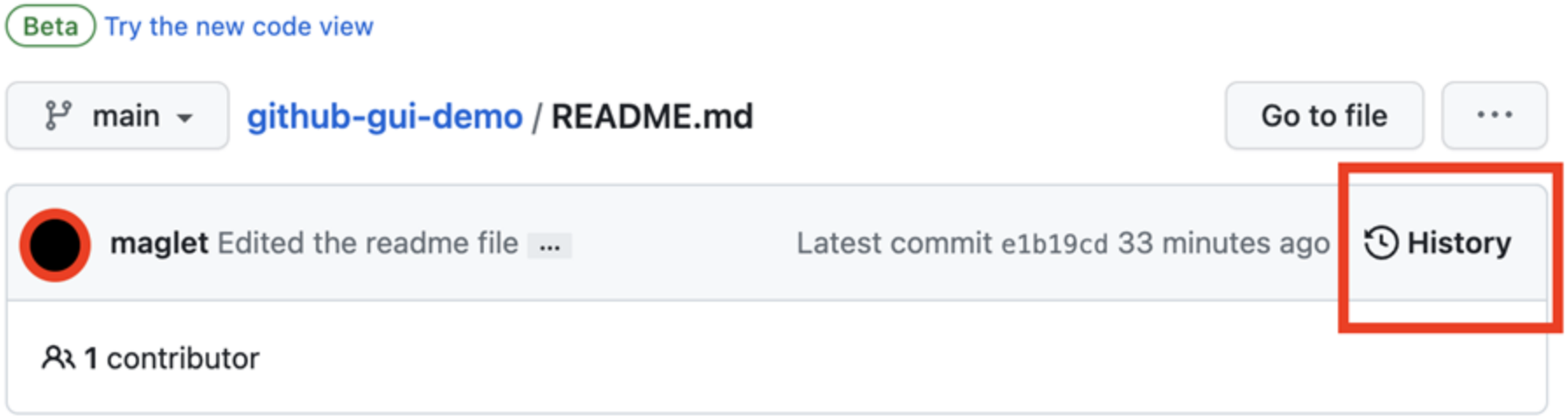
Blame

See when each line of a file was edited last

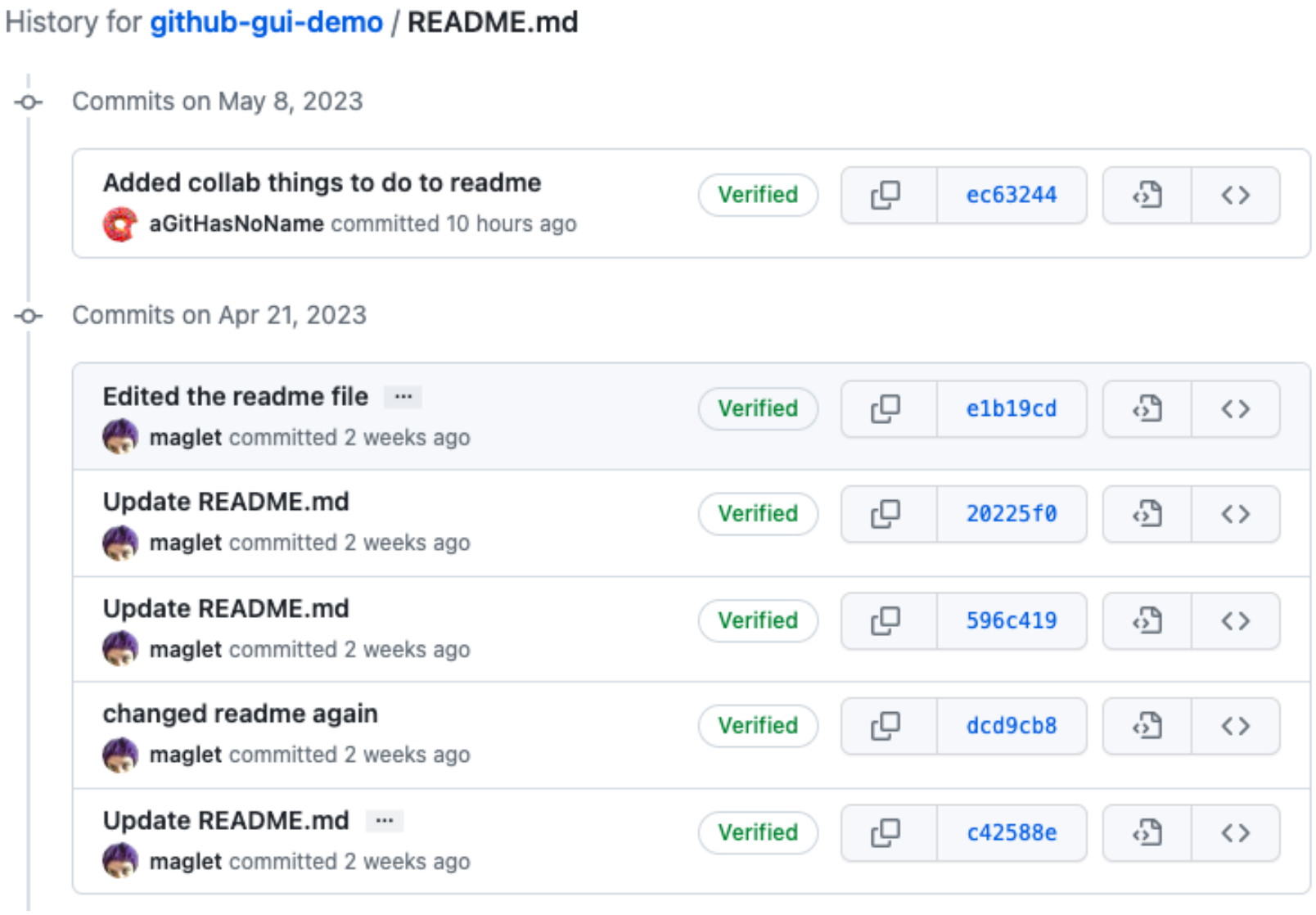


View File History

See all commits that have been made

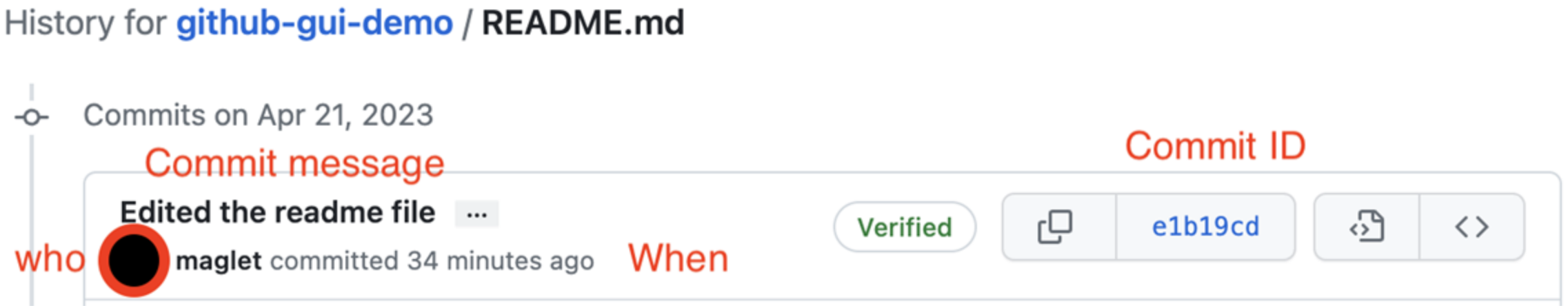


View File History



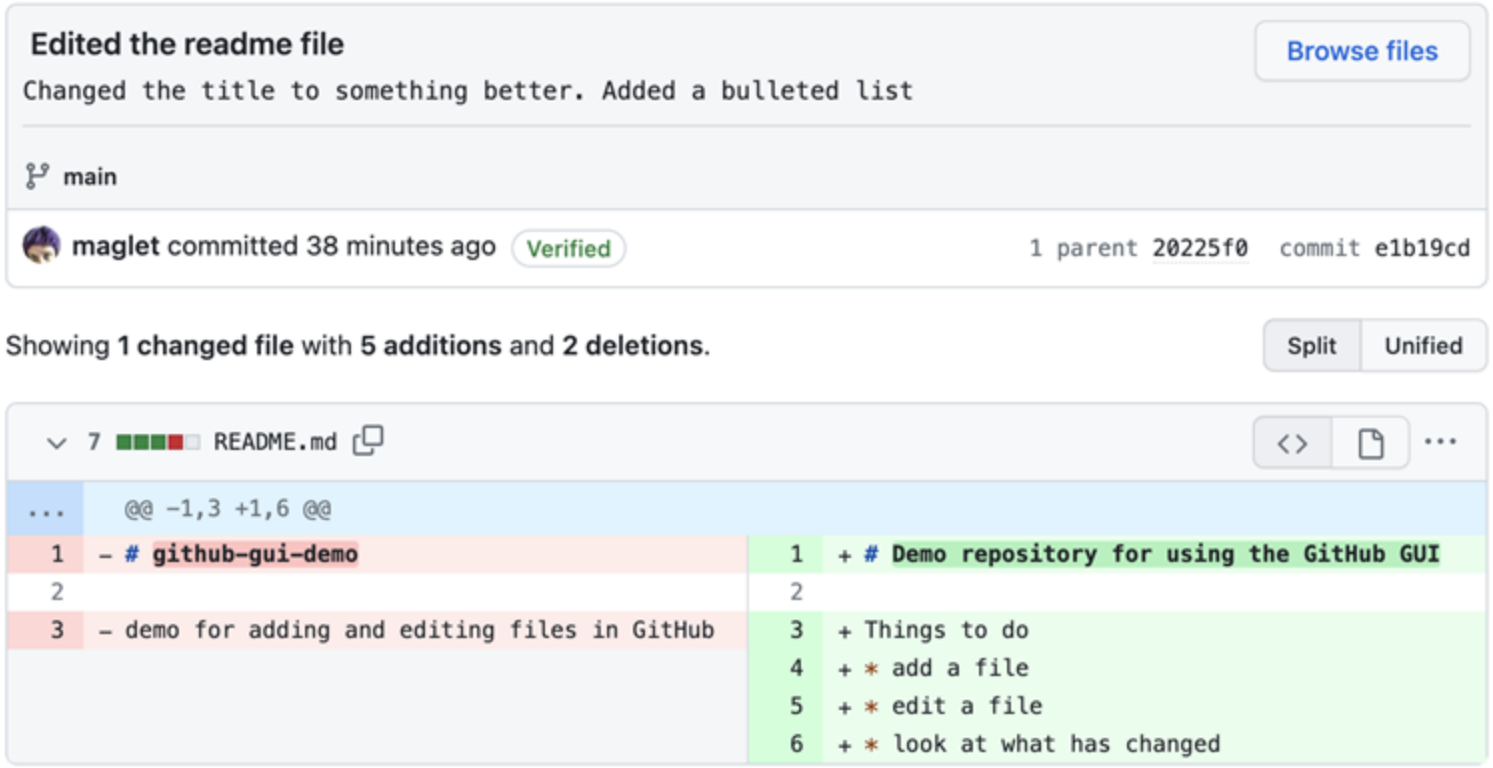
View File History

See who made the commit, when, and the commit message



diff

See what changed in the file line by line

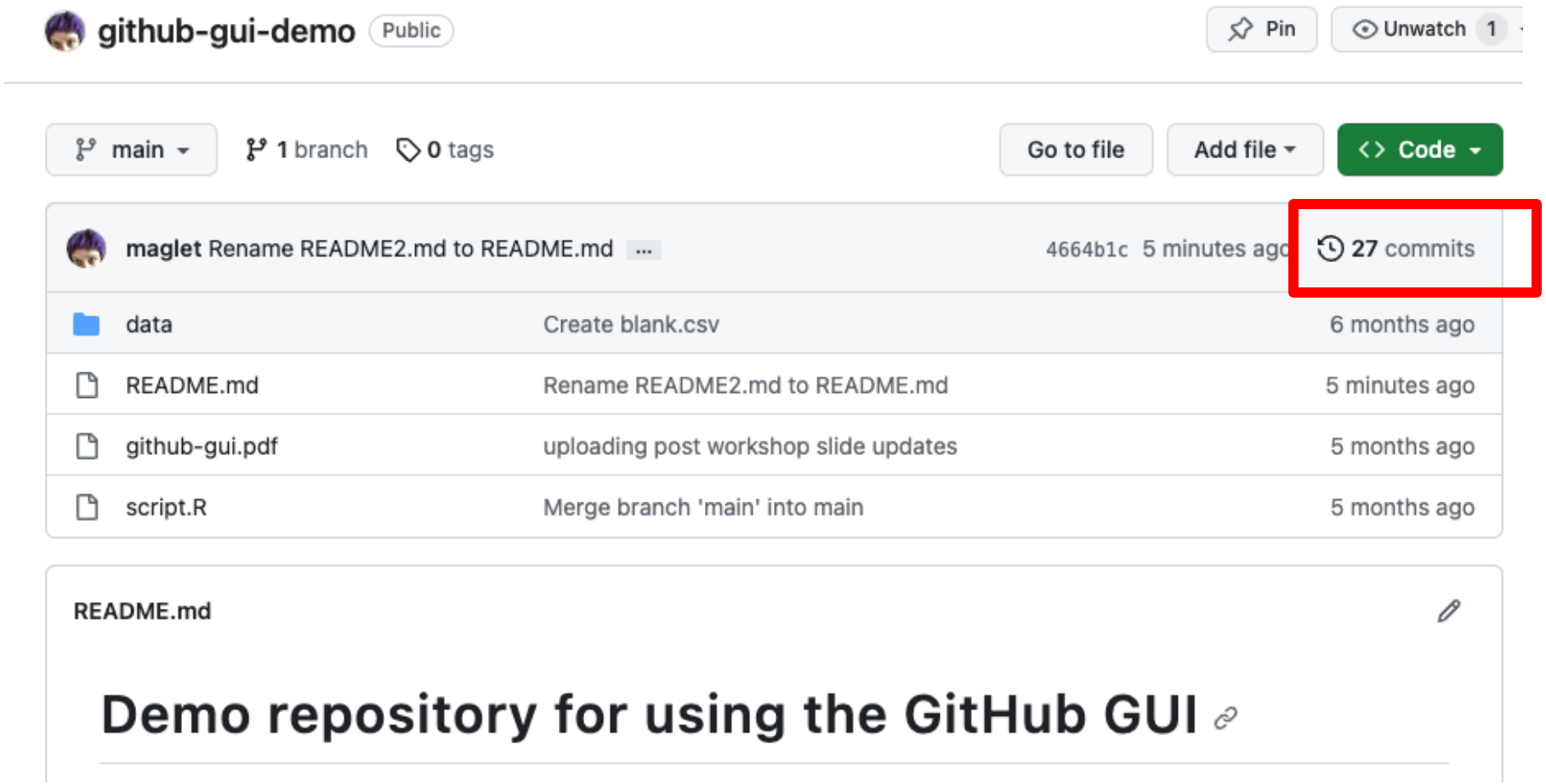


Red = deleted

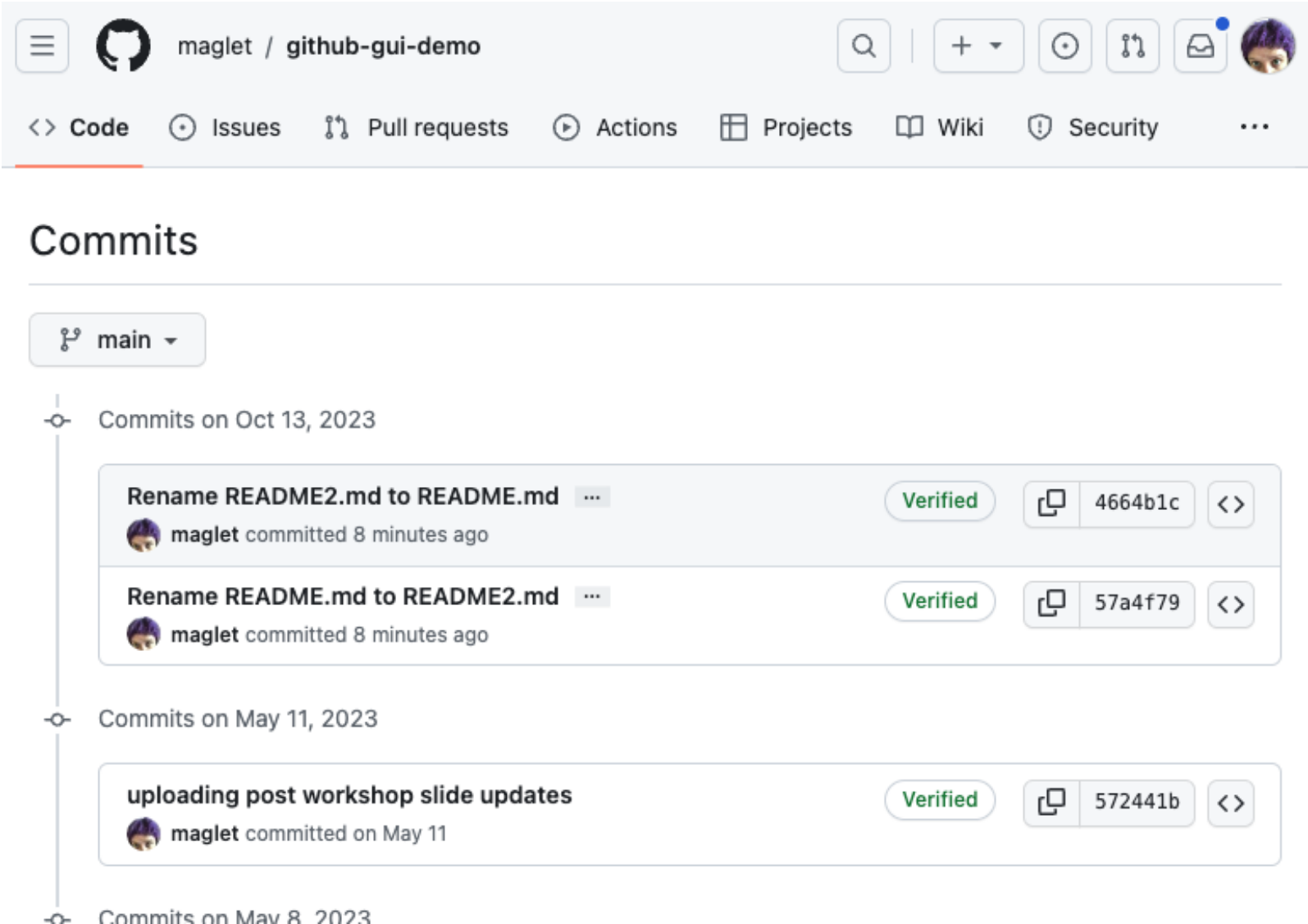
Green = added

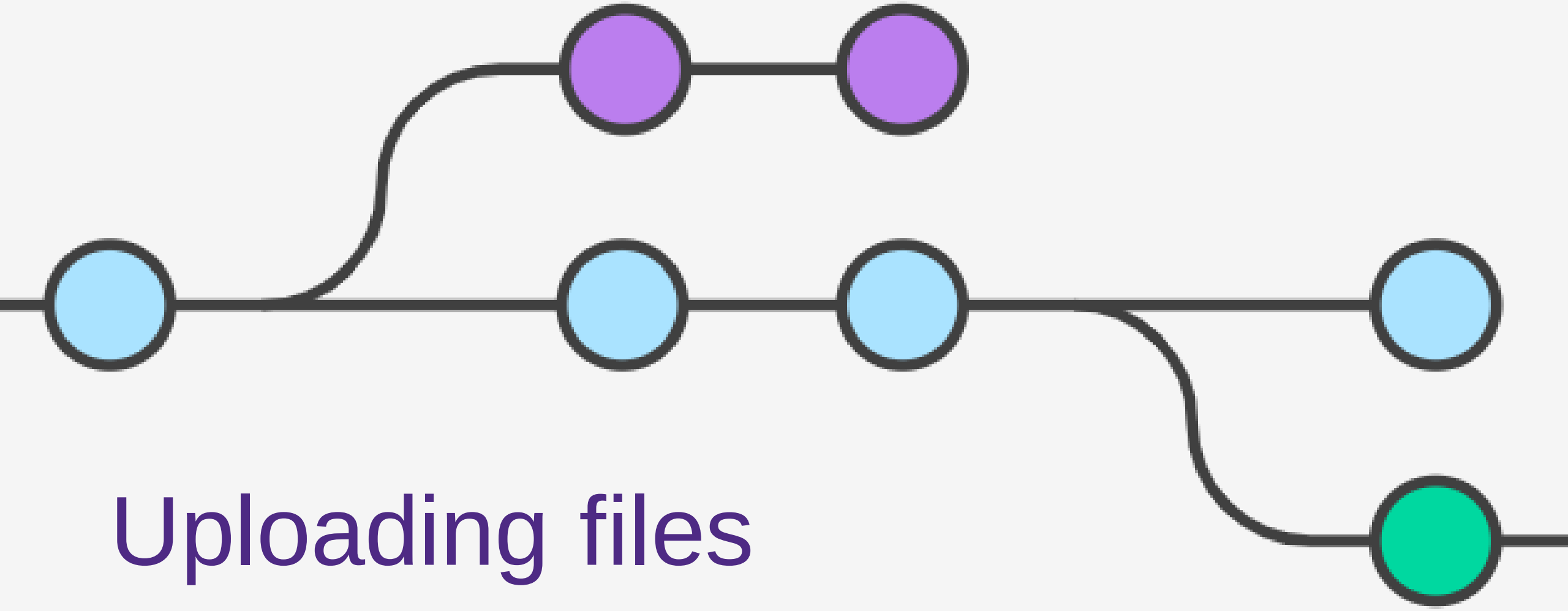
View Repo History

From the main page



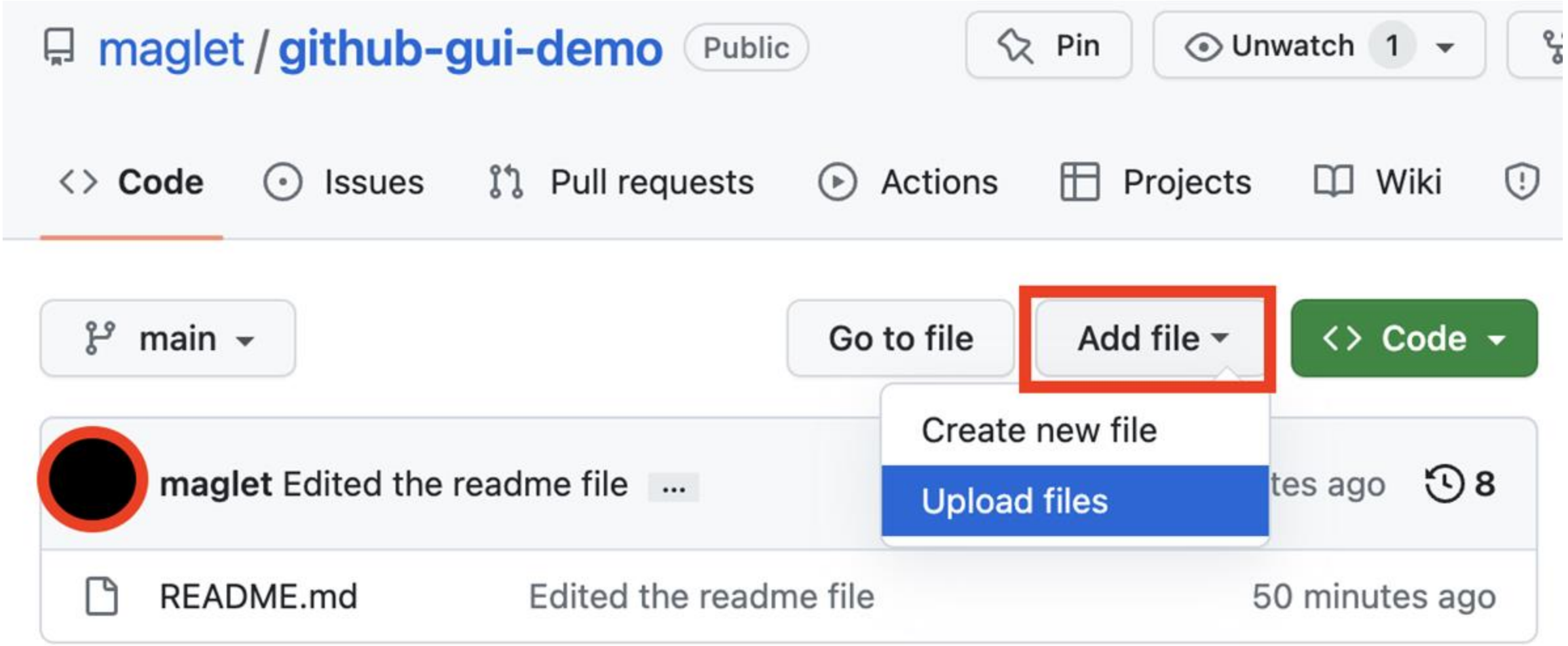
View Repo History





Uploading files

Add a file from your computer



Choose files

github-gui-demo /

Drag files here to add them to your repository

Or choose your files

Commit

Commit changes

Added an R script

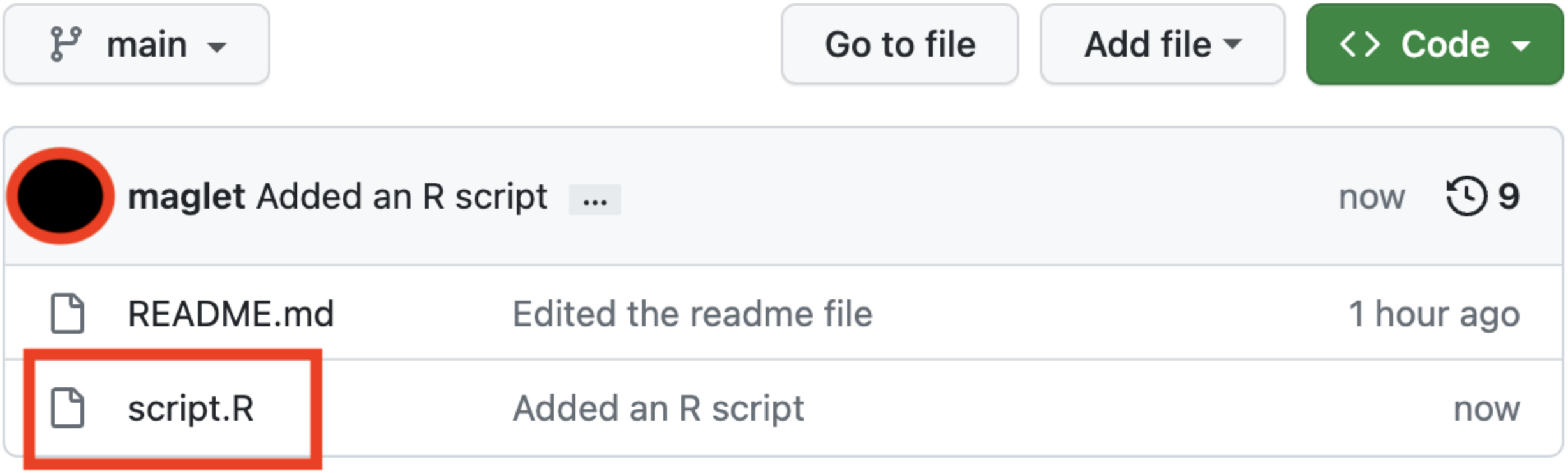
Add an optional extended description...

◉ Commit directly to the `main` branch.

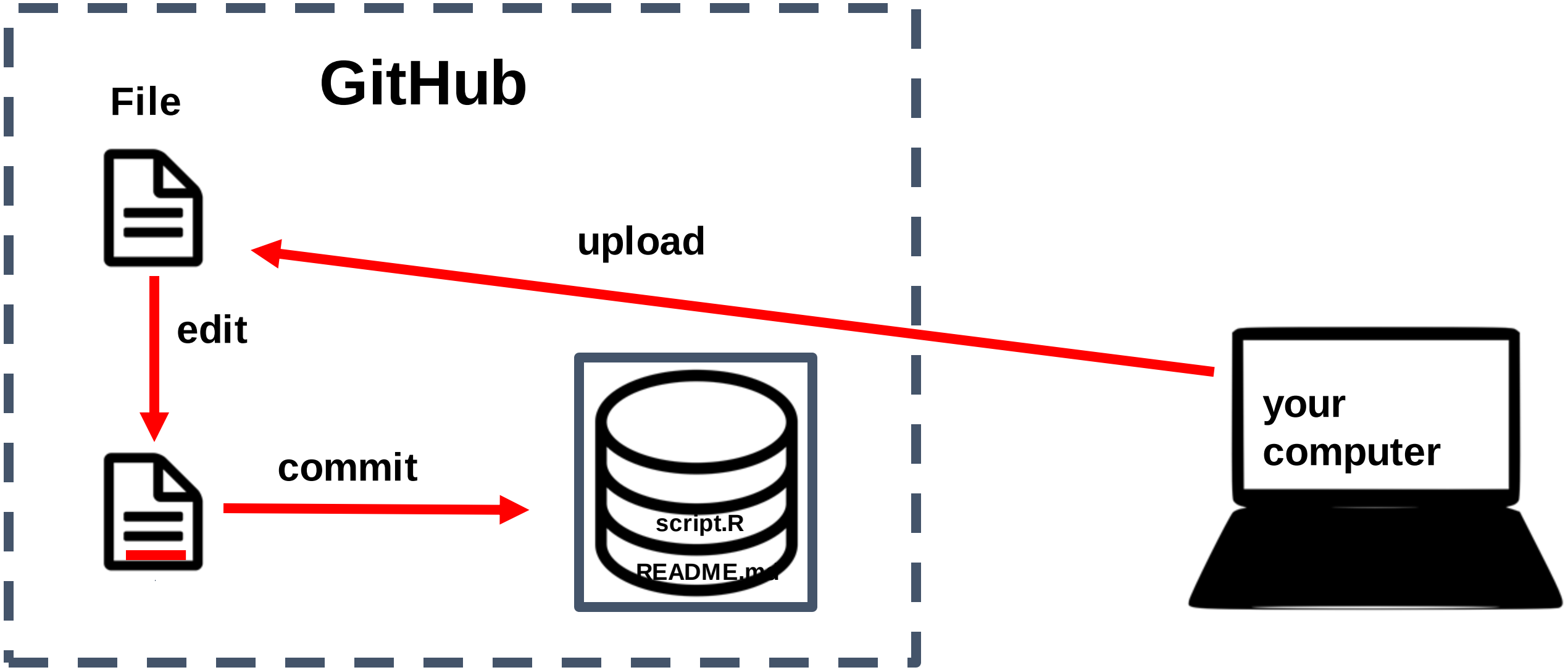
○ Create a **new branch** for this commit and start a pull request. Learn more about pull requests.

Commit changes | Cancel

File was added

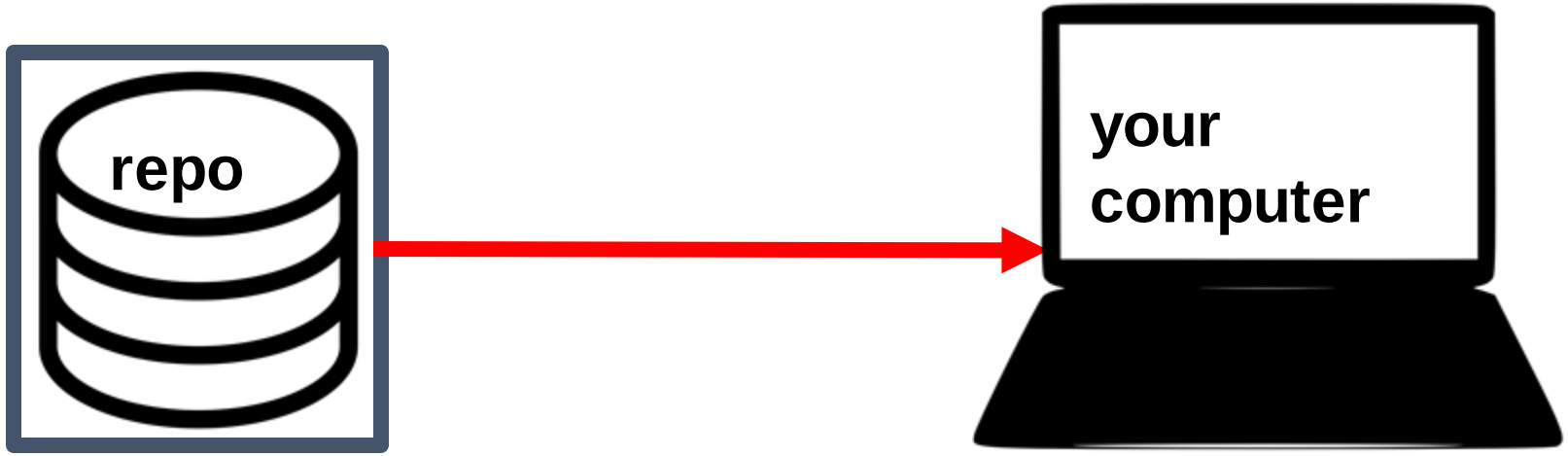


What we've done so far



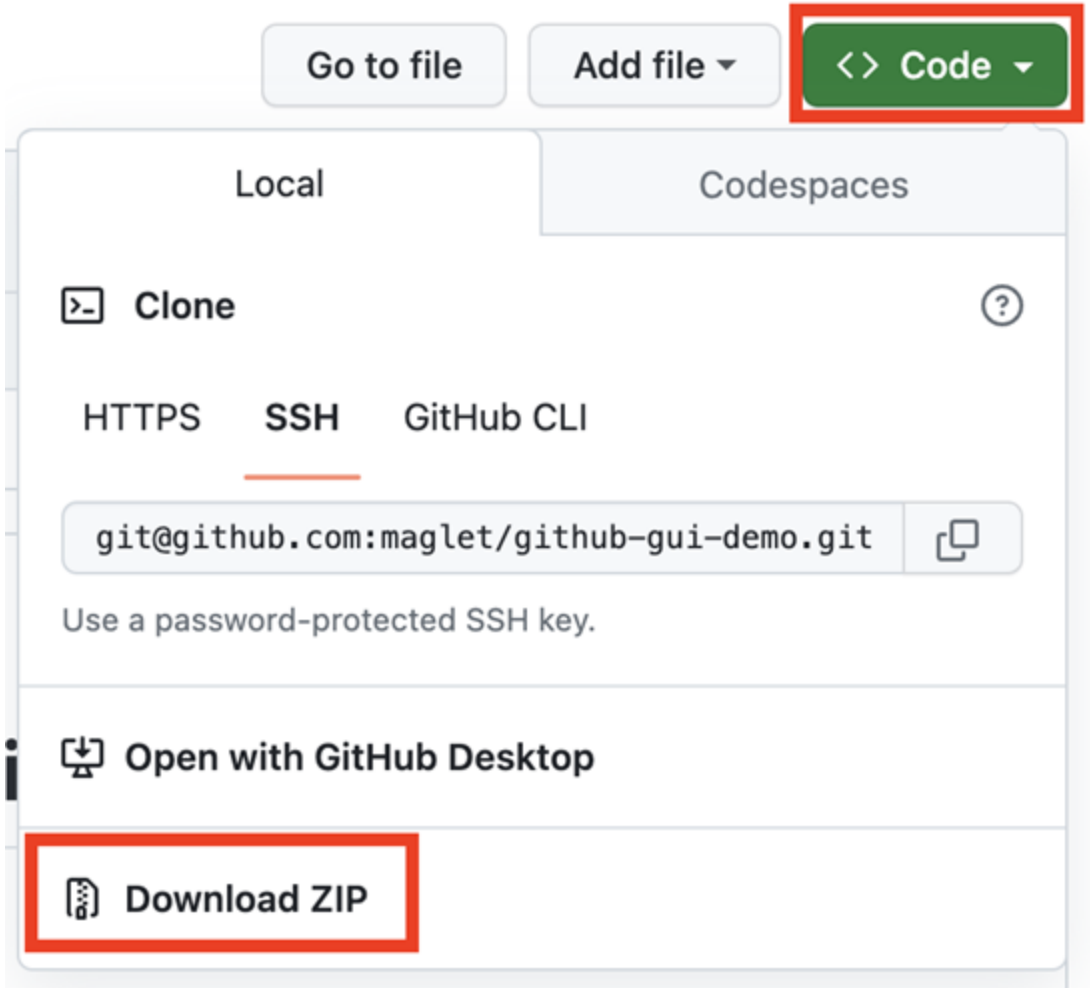
Running code

- **Copy and paste** - take the lines of code you need to run locally
- **Download** - download most recent version of files
- **Clone** - downloads the whole repository to do version control on your computer



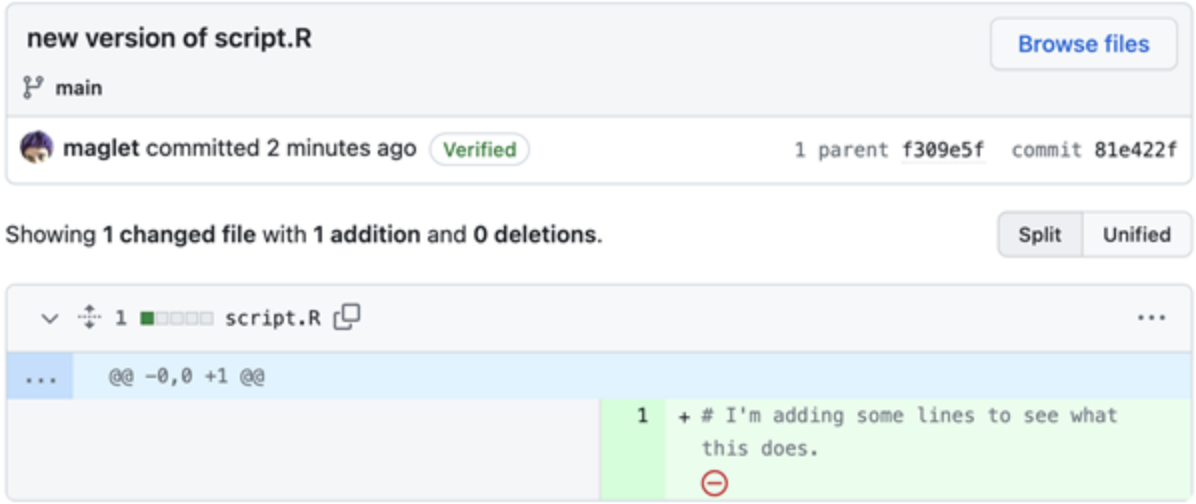
Download

- Click the green code button
- Click download ZIP
- Extract files using a program on your computer
- Open your files

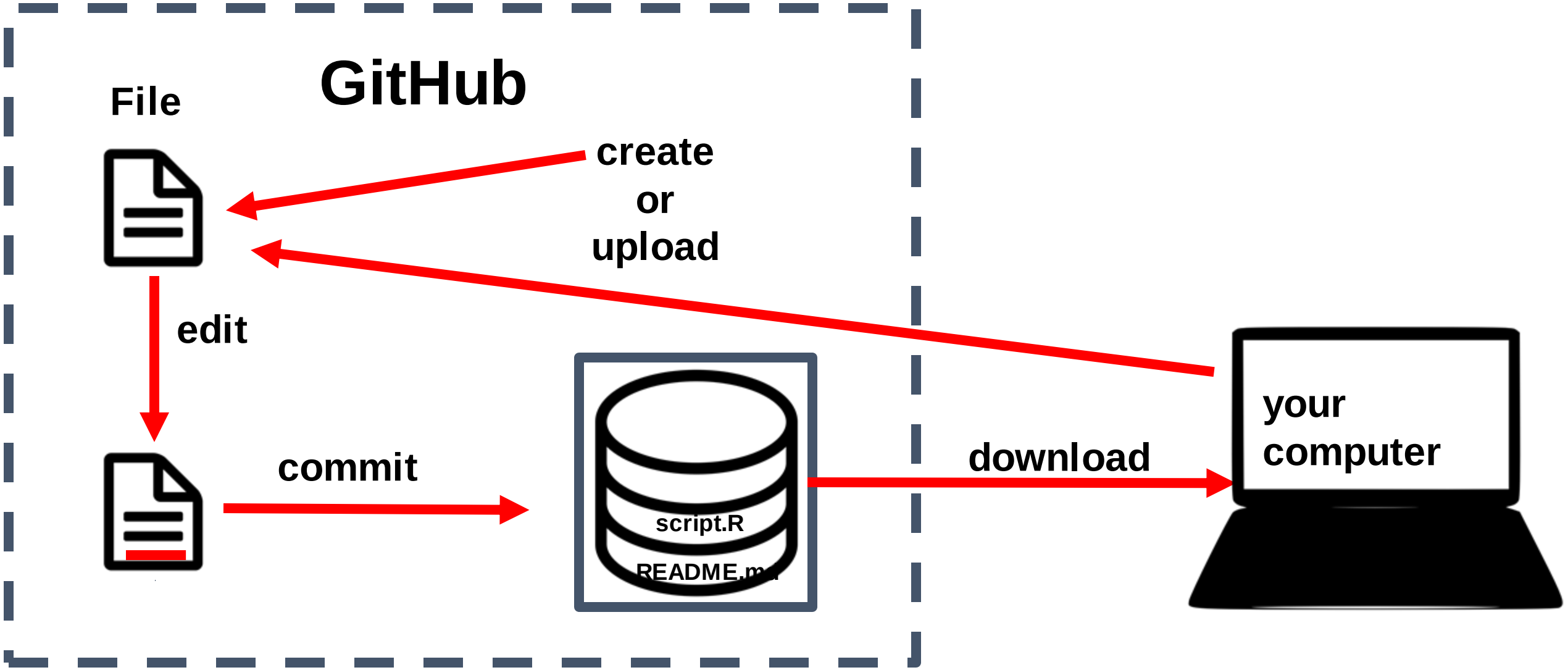


Save changes

- Add file > upload file
- Select the file you changed
- Commit with a message
- File is updated



GitHub workflow overview



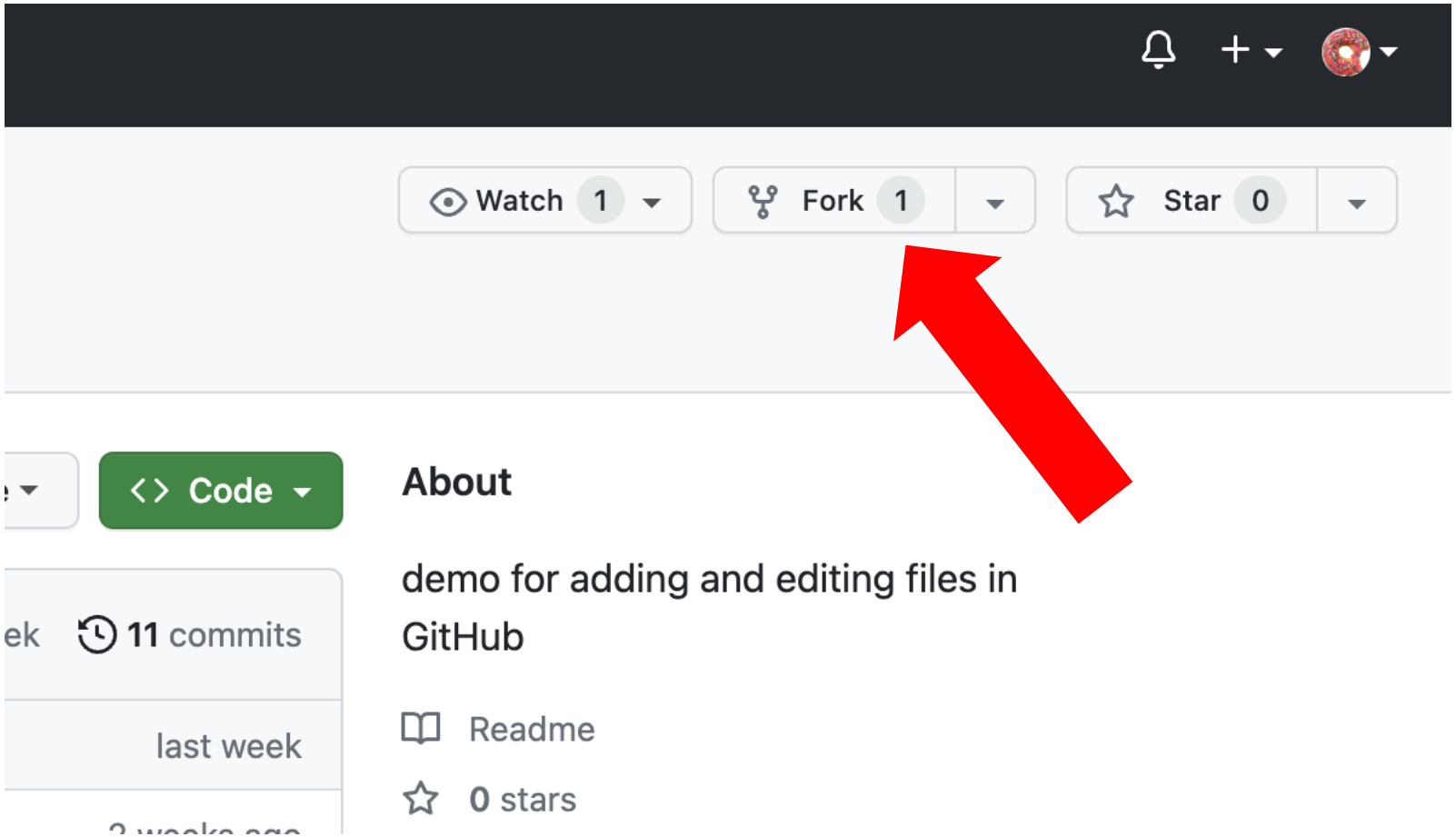

Collaboration

SO MUCH JARGON with Git

STEP ONE: CREATE A **FORK**

Create a Fork from your friend's repo (AKA make a copy of the repo in your own account).

You can now play around with your version without messing up your friend's.

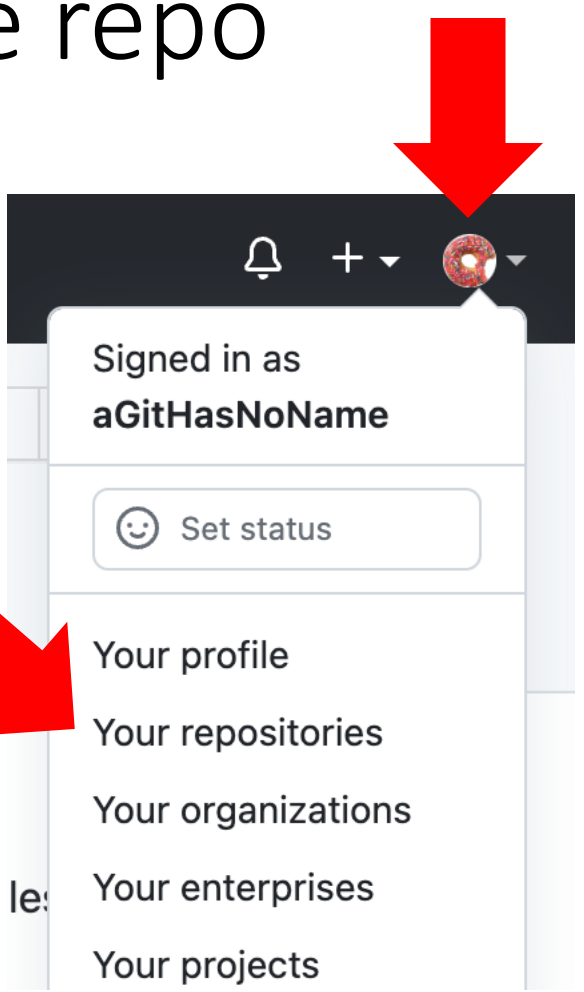


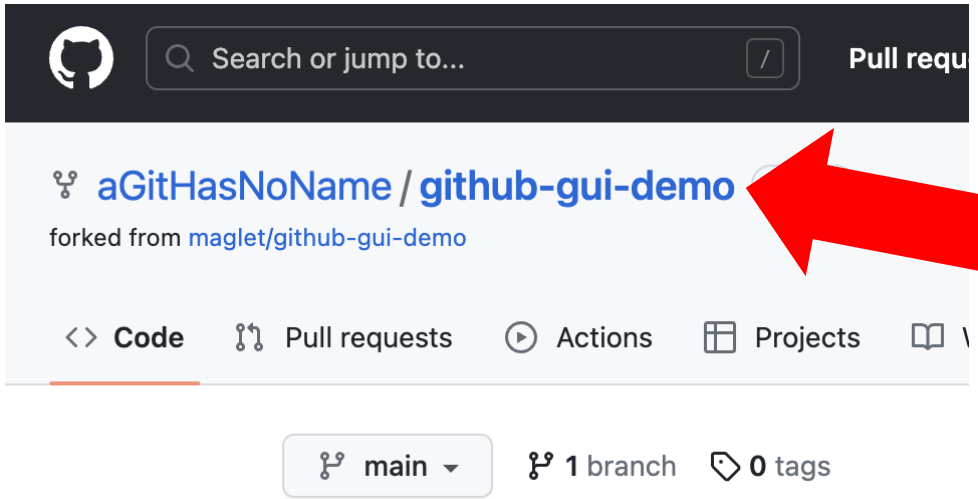
Do not create a **FORK** from a FORK

If your friend's repo is actually a fork of someone else's repo, the following instructions for collaboration will not work. If you want to contribute to the original repo, you need to create a fork from the original repo.

STEP TWO: **OPEN** your version of the repo

Click on your account in the top right corner of your screen. Then click on Your repositories. Choose the forked repo.



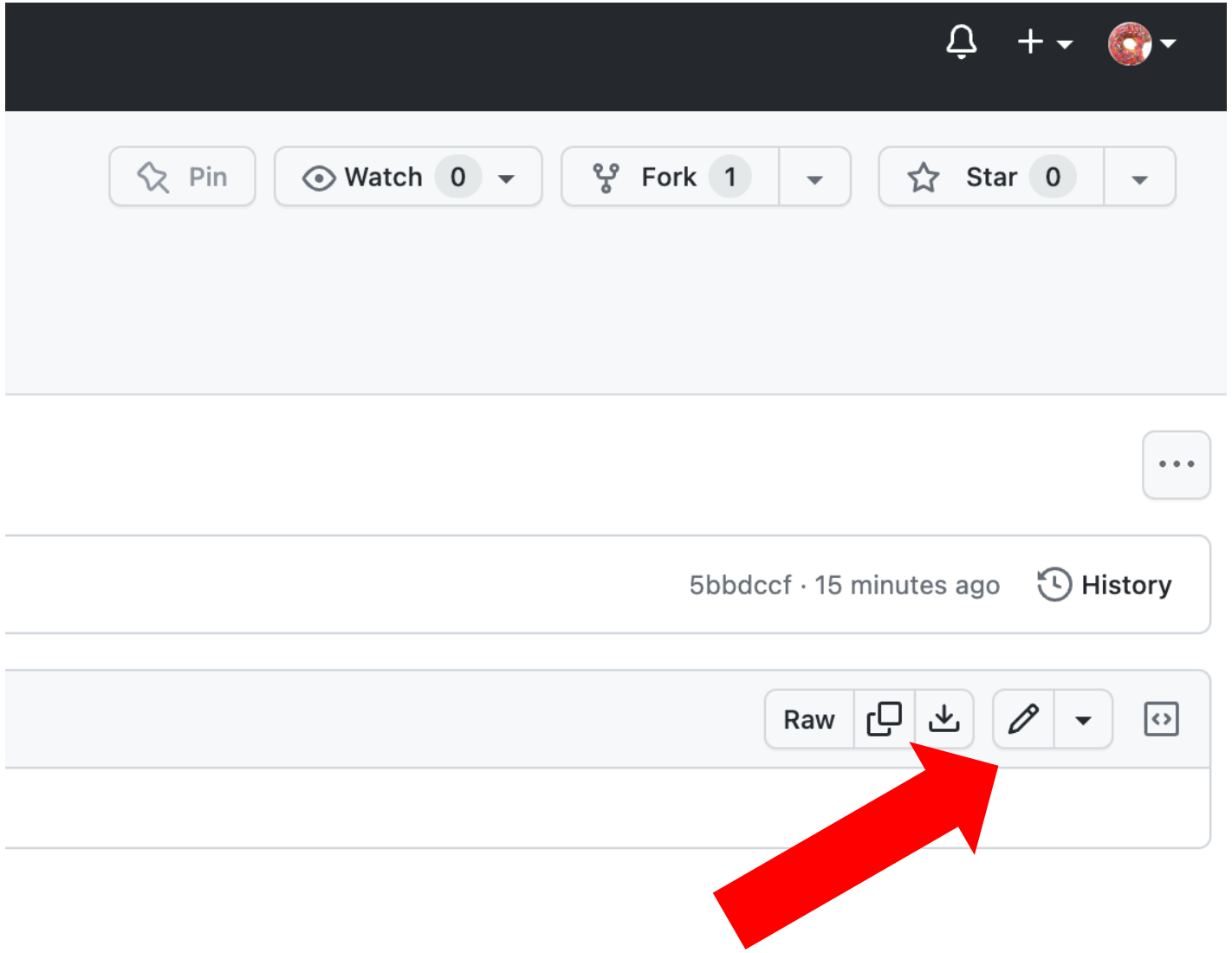


You can see it's your version of the repo in the top left of your screen.

STEP THREE: **EDIT** a file

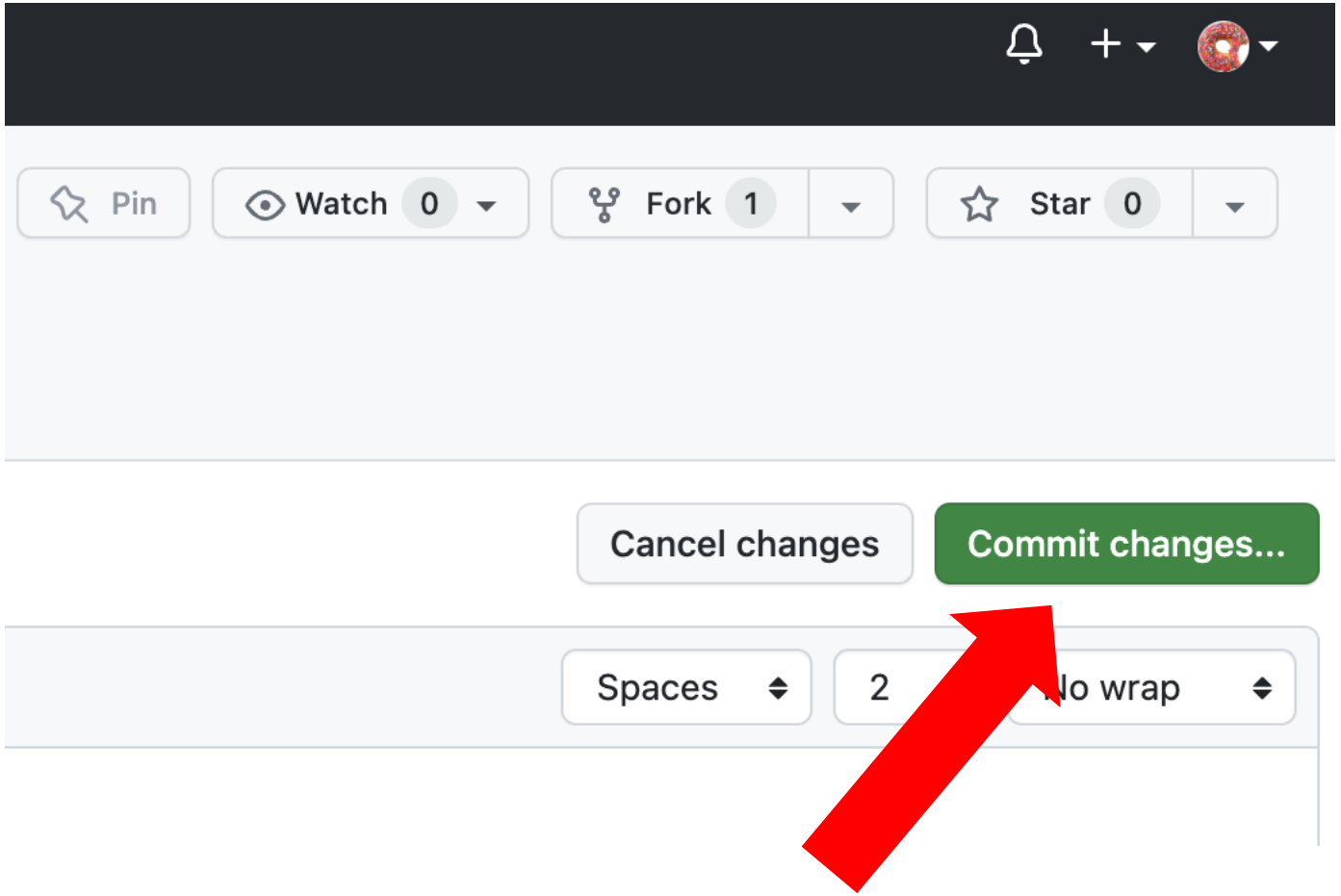
Open a file in the repo. Click on the pencil icon to Edit the file.

You can make direct changes or paste in new code you've been working on.



STEP FOUR (a): **COMMIT** your changes

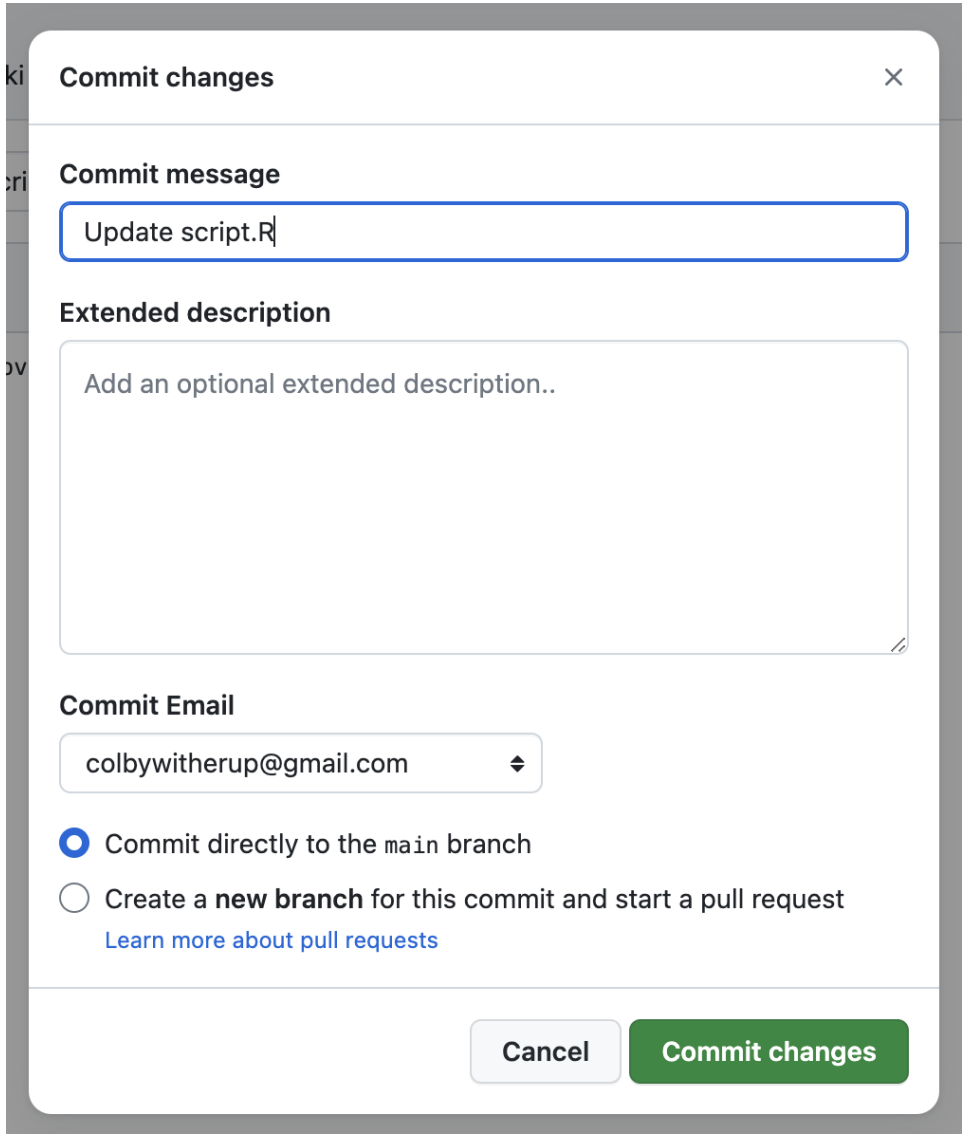
The button will light up once you've made a change to the file.



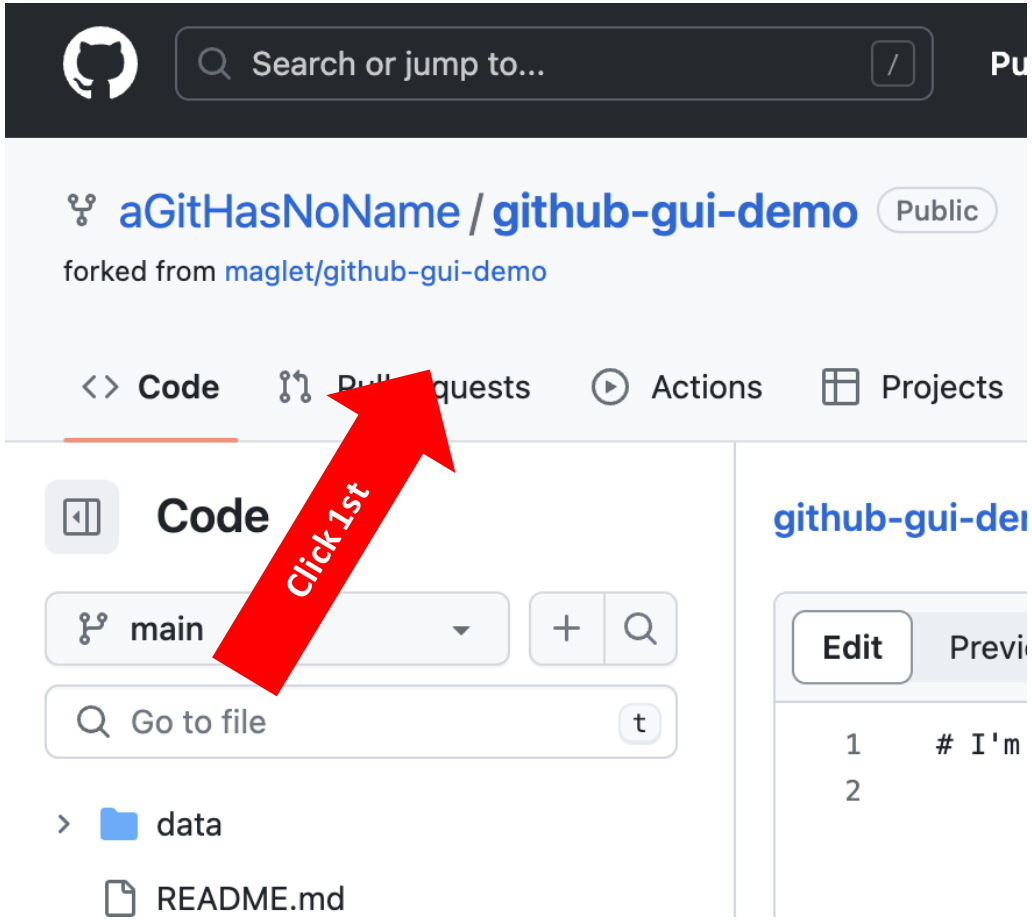
STEP FOUR (b): **COMMIT** your changes

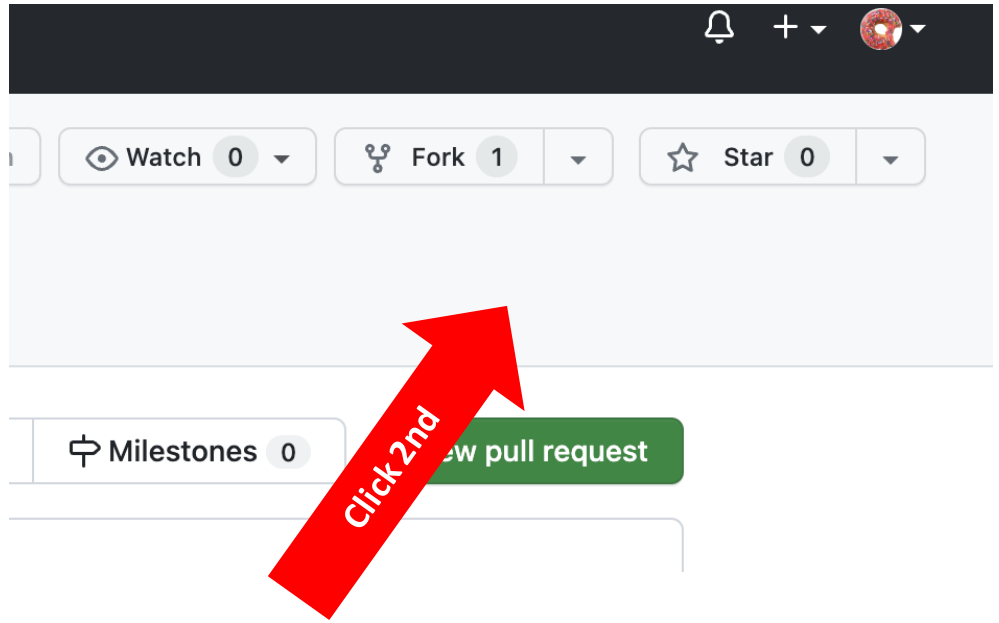
Name your commit.

You can commit directly to the **main branch** (that's the main branch in YOUR version of the repo). This lets you keep working on this file or on other files too before sending all your changes to your friend.



STEP FIVE (a): Create a **PULL** request





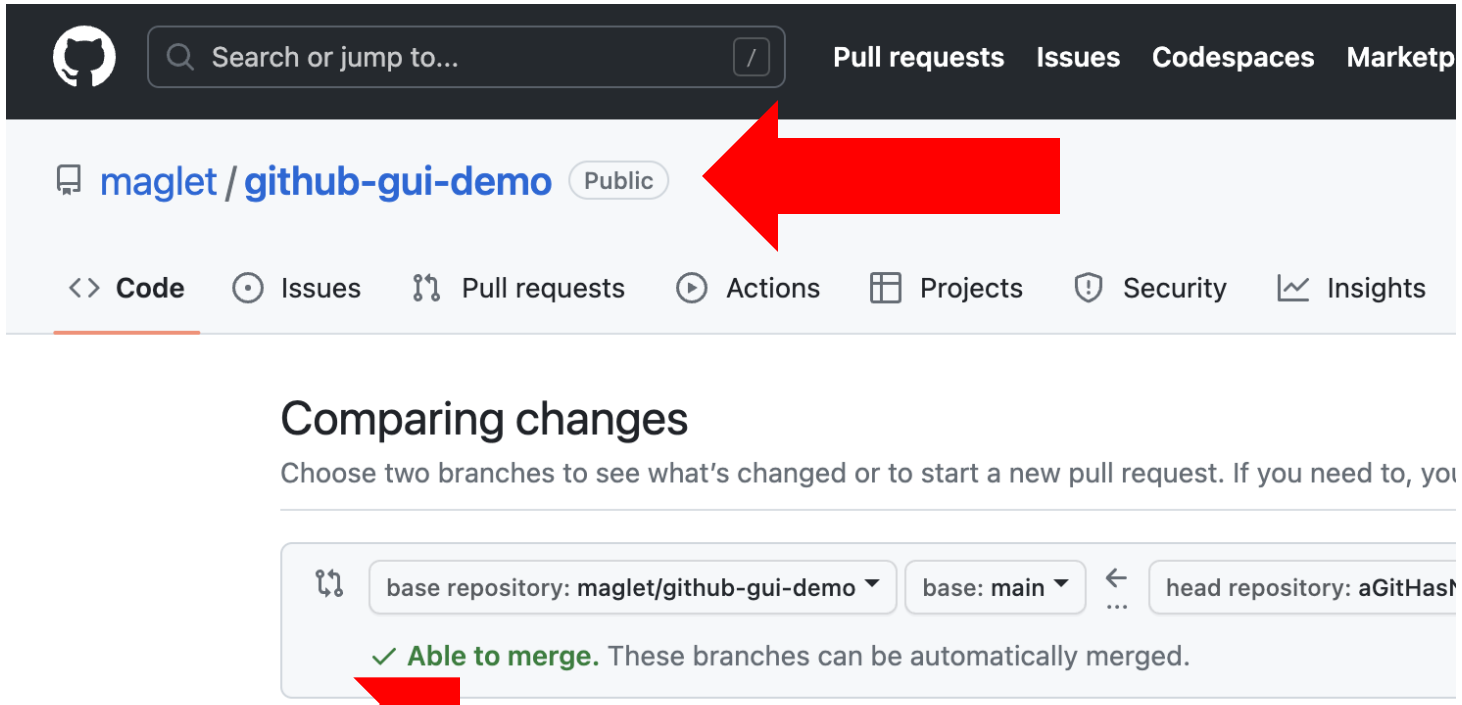
PUSH

- AKA “**Create a pull request**” when working with someone else’s repo
- Make changes (AKA **commits**) to your version of the repo and ask them to accept those changes into their version of the repo
- *Remember, your friend won’t lose their previous version if they accept your changes – all versions are saved!*

PULL

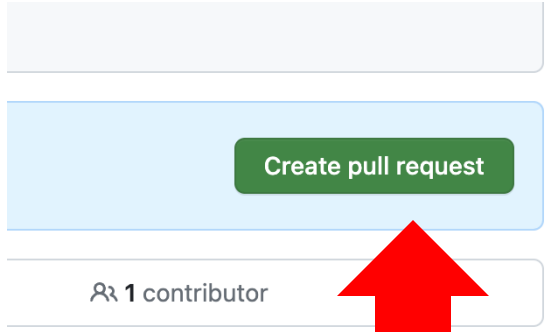
- Like ”Accept all changes” in Microsoft Word when you let someone else review and make suggestions to your doc

STEP FIVE (b): Create a **PULL** request



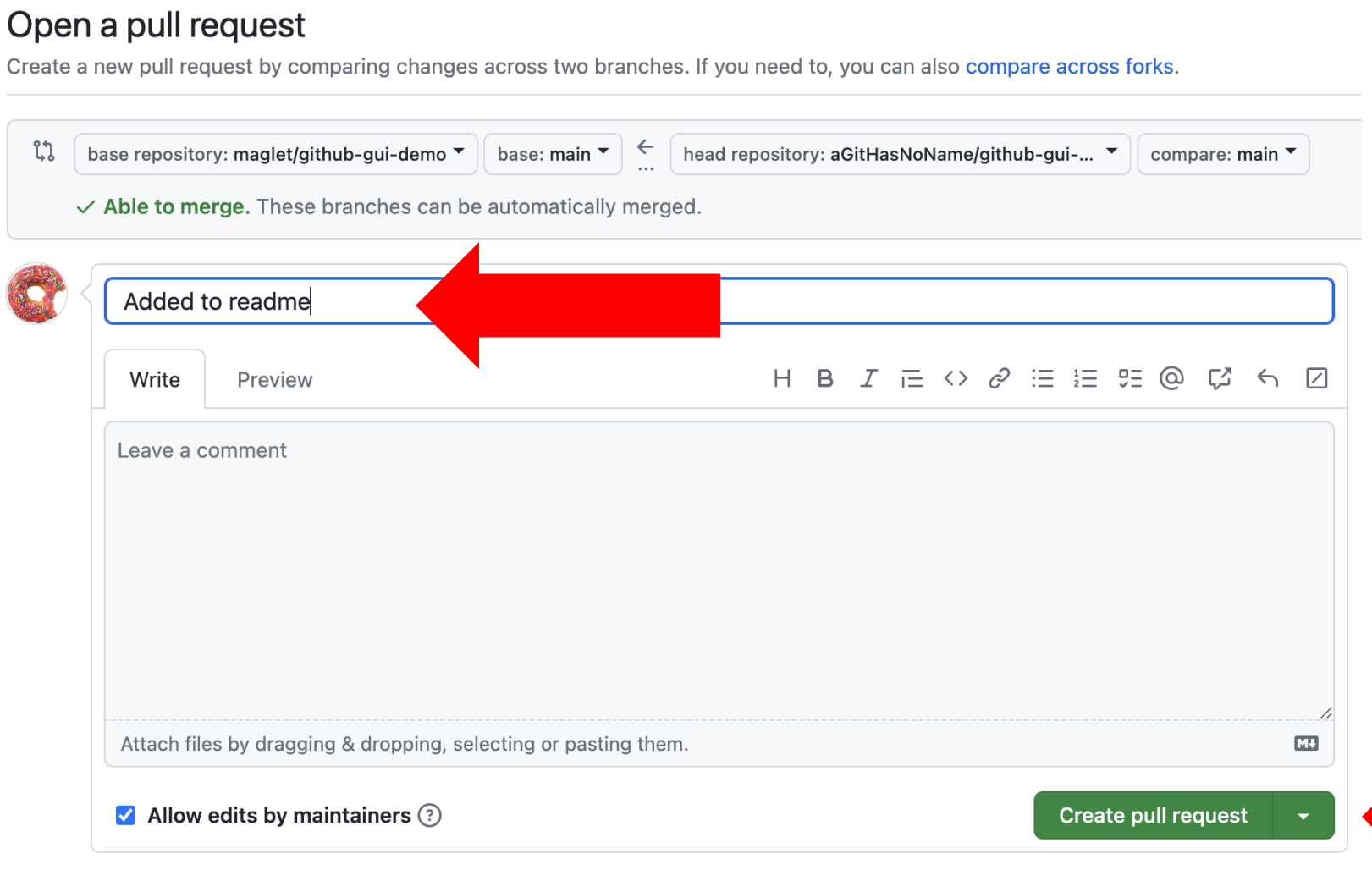
It will now list your friend's version of the repo.

If it says "Able to merge", click the green button.



If it doesn't say "Able to merge", that means your friend has made changes since the last time you merged. Jump ahead to Step Five (d) for help.

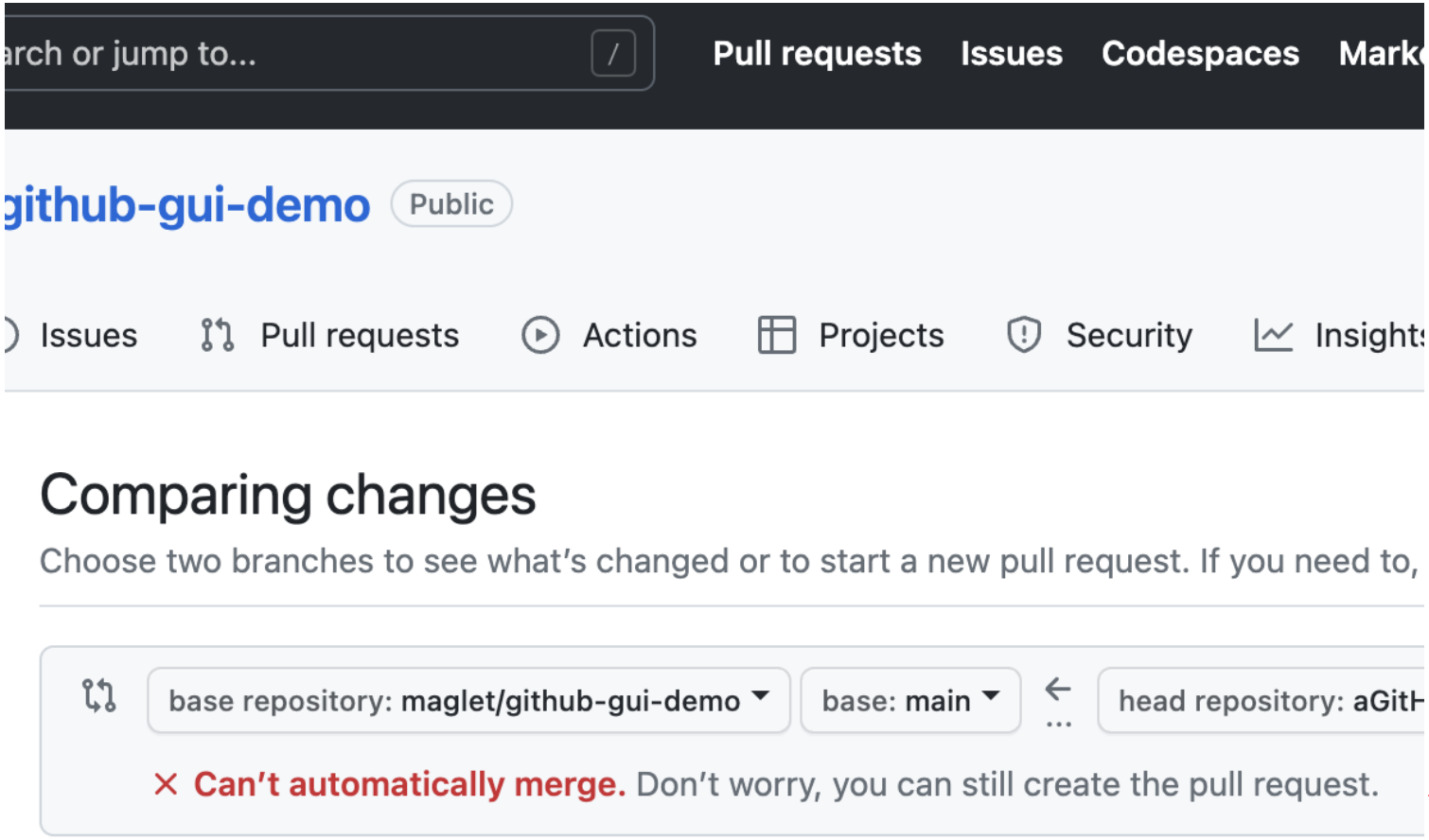
STEP FIVE (c): Create a **PULL** request



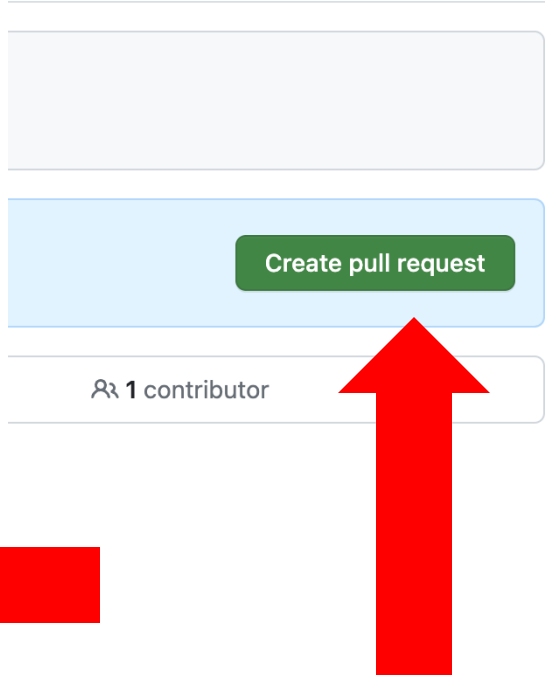
Give the pull request a name. You can also write a comment to your friend.

Click the green button.

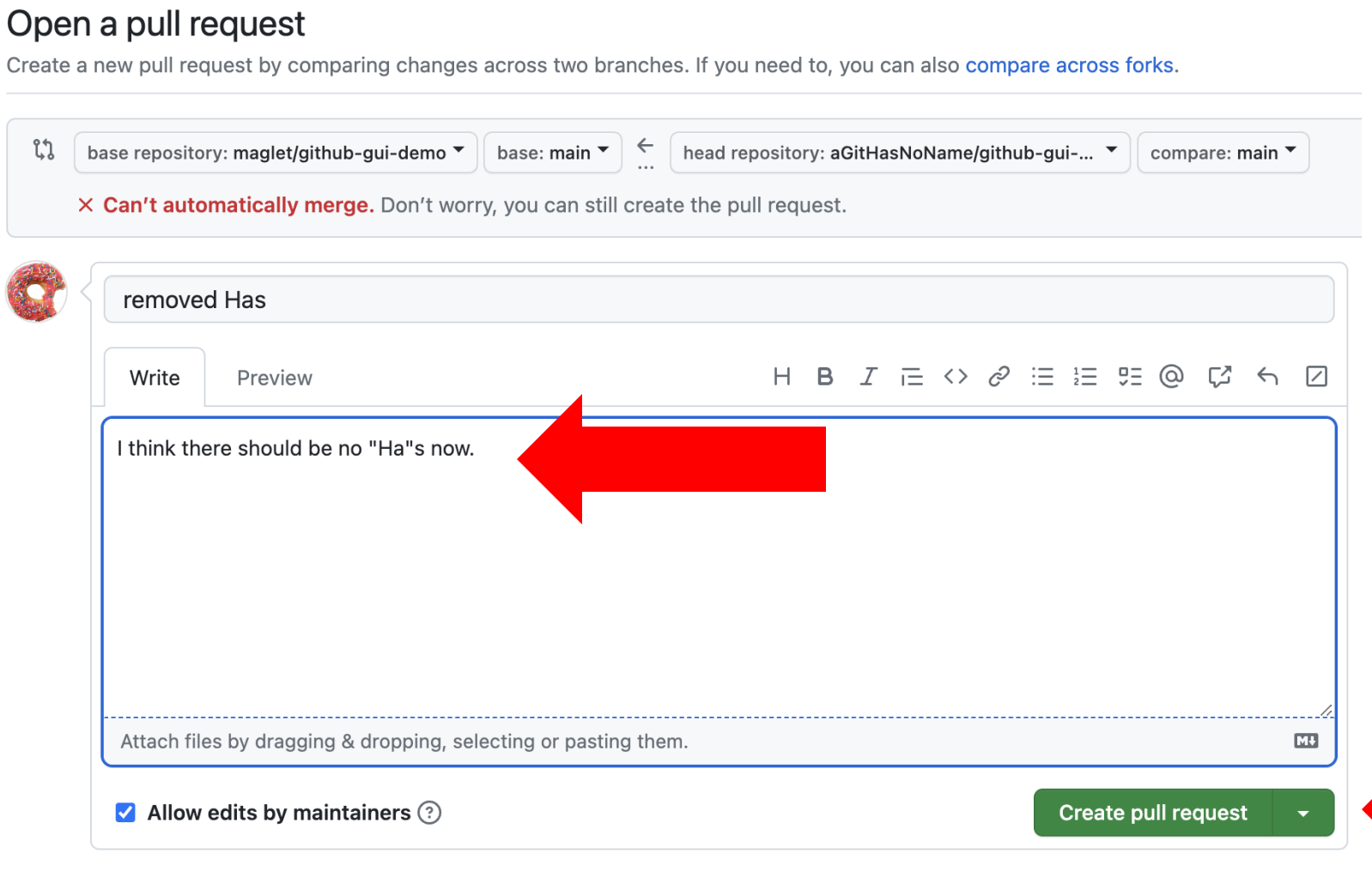
STEP FIVE (d): **HELP!** Can’t automatically merge



Click the green button anyway.



STEP FIVE (d): **HELP!** Can’t automatically merge



Make your case to your friend about why your changes are waaaay better than their changes.

Click the green button.

STEP FIVE (e): **WAIT**

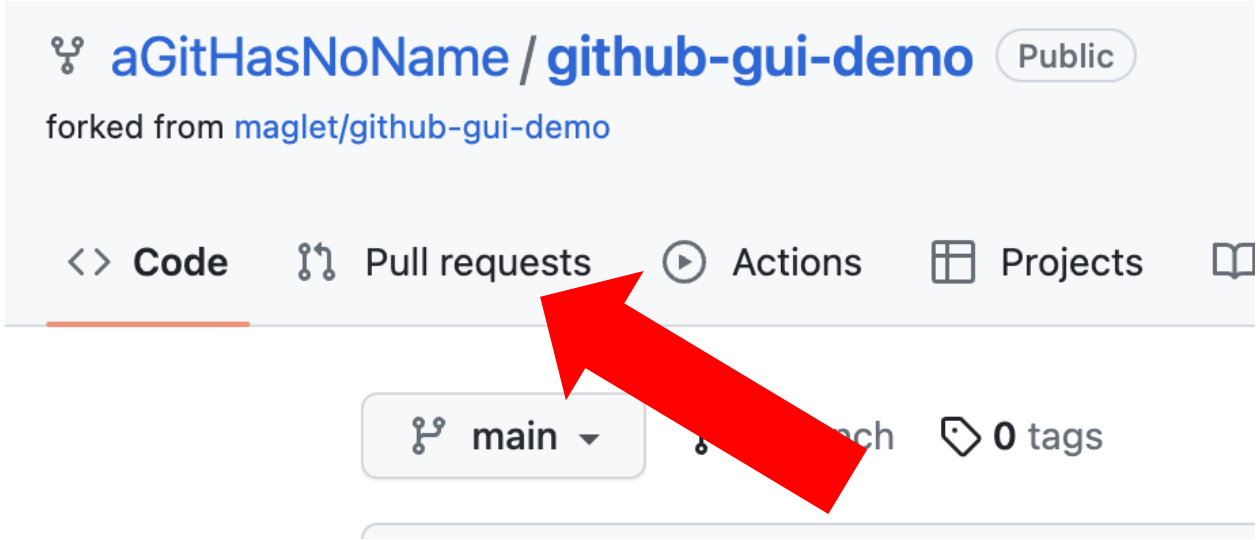
Wait for your friend to accept your pull request.

IMPORTANT: If you make any new commits before your friend PULLS, they will go into the same pull request and get pulled along.

NOTE: "Close a pull request" means CANCEL the pull request.

How to accept a pull request

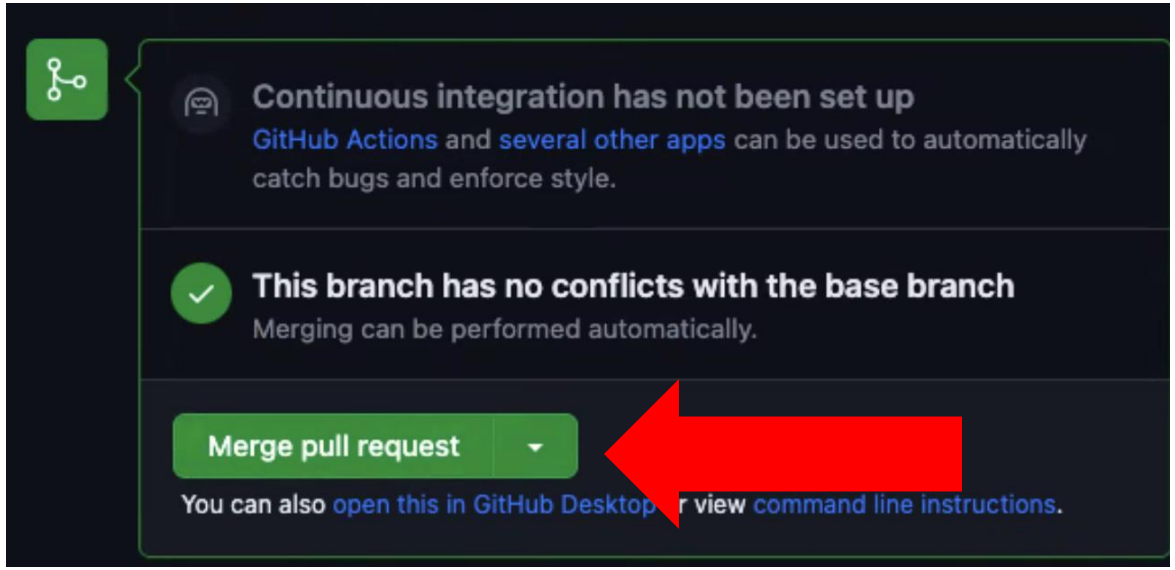
STEP ONE: Open the pull request



Click on "Pull requests" in your repo.

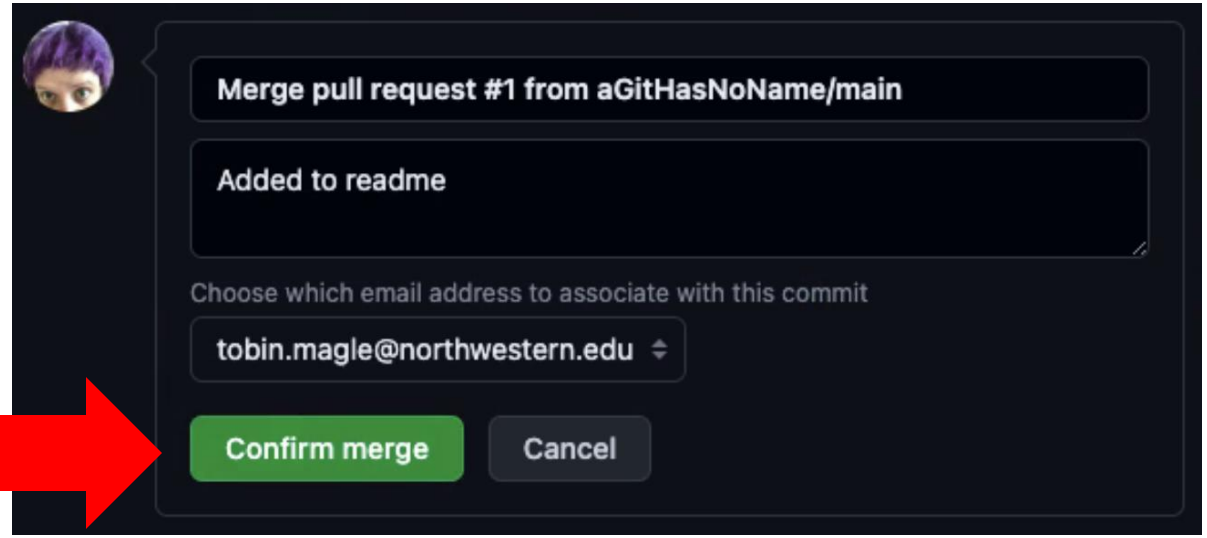
Choose the appropriate pull request from the menu.

STEP TWO: **MERGE** your friend's changes into your repo



Click on "Merge pull request", then "Confirm merge"

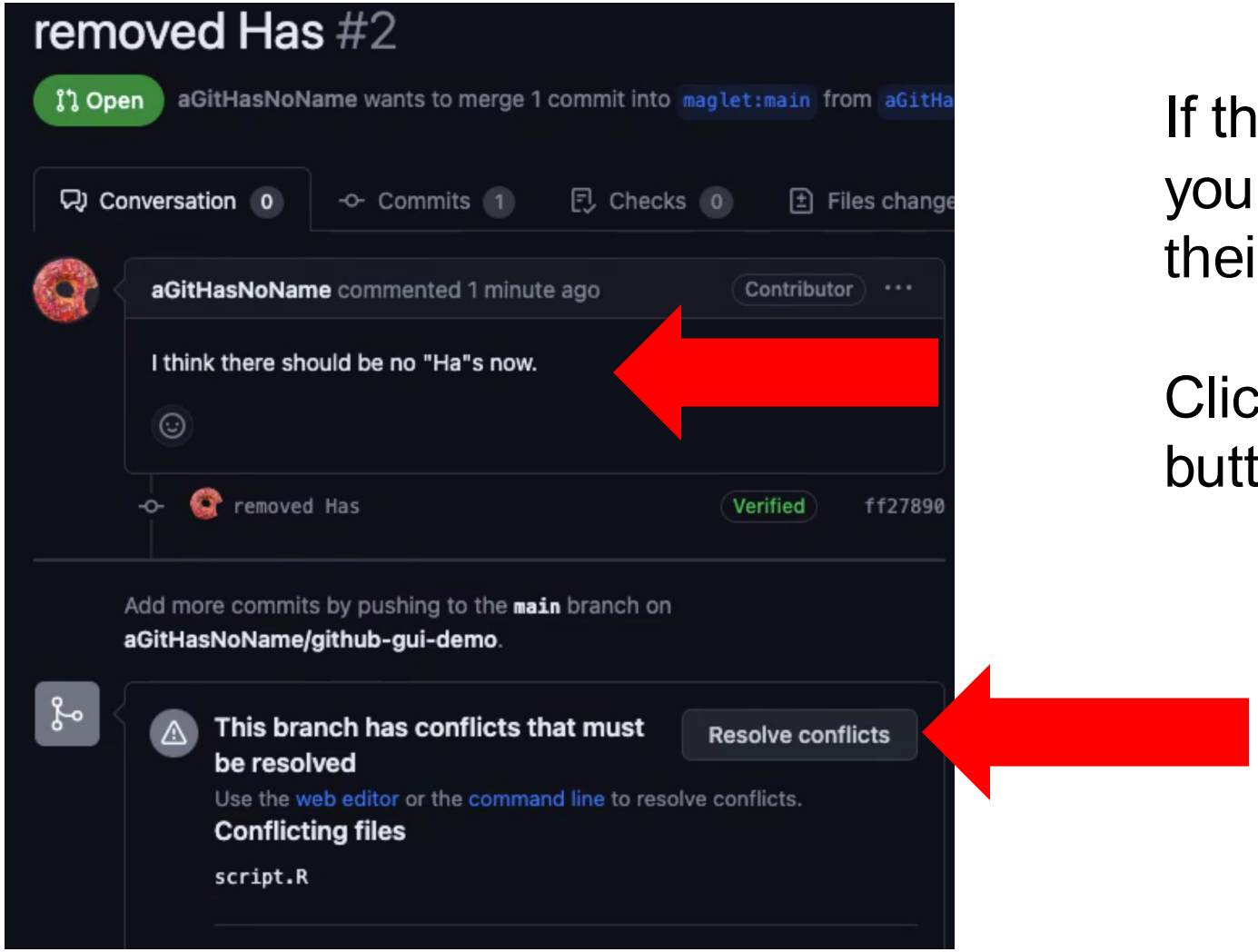
NOTE: You can reject a pull request by "Closing" it.



STEP TWO: **MERGE** your friend's changes into your repo

It worked!

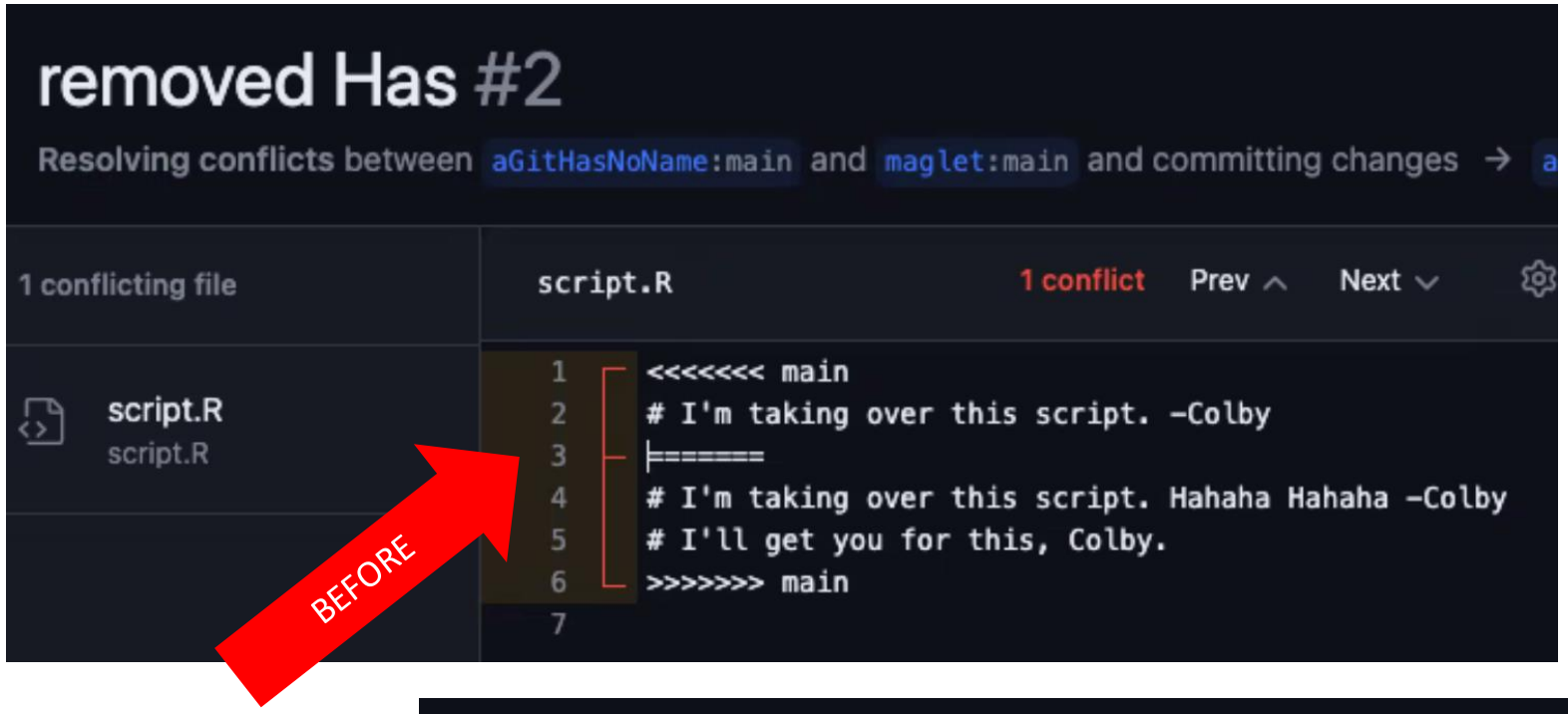
STEP TWO (b): RESOLVE any conflicts



If there's a conflict, you'll see your friend's explanation of their changes.

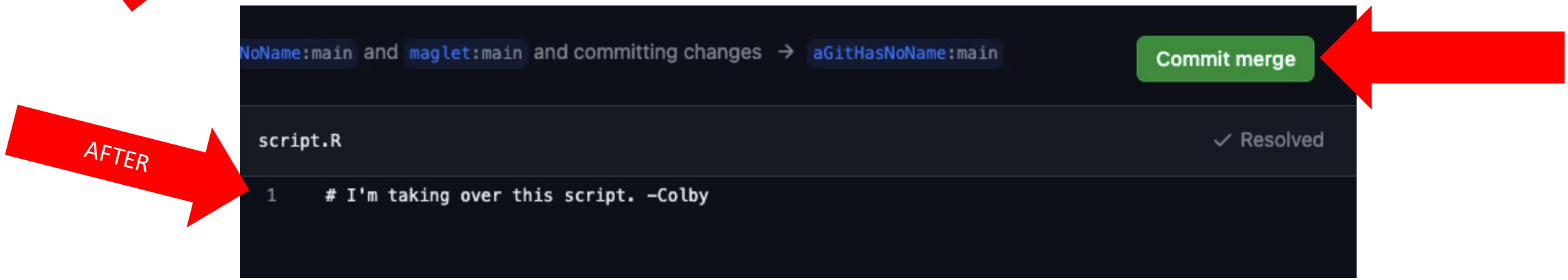
Click the "Resolve conflicts" button. It's a little hidden.

STEP TWO (b): RESOLVE any conflicts

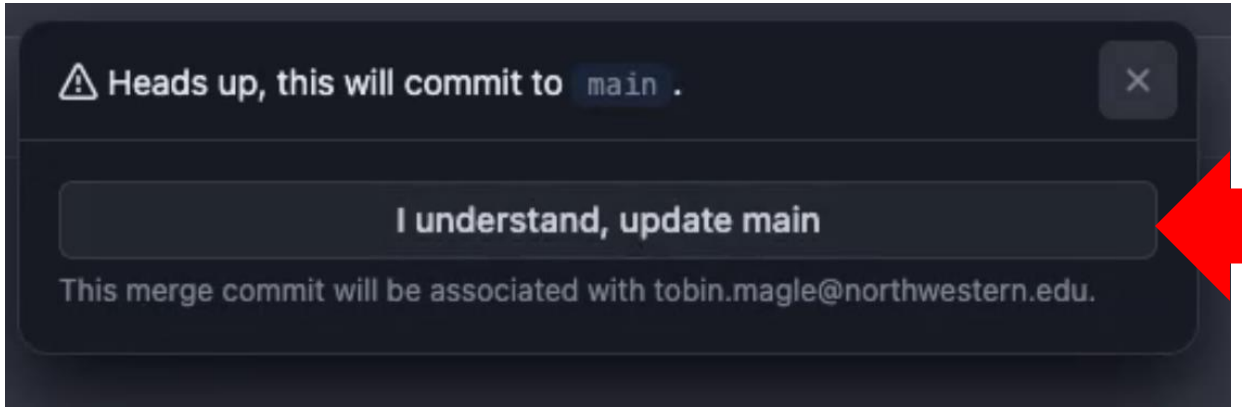


View the conflict. Decide what it should really be and edit it directly in the box.

Once it looks correct, click the green button.

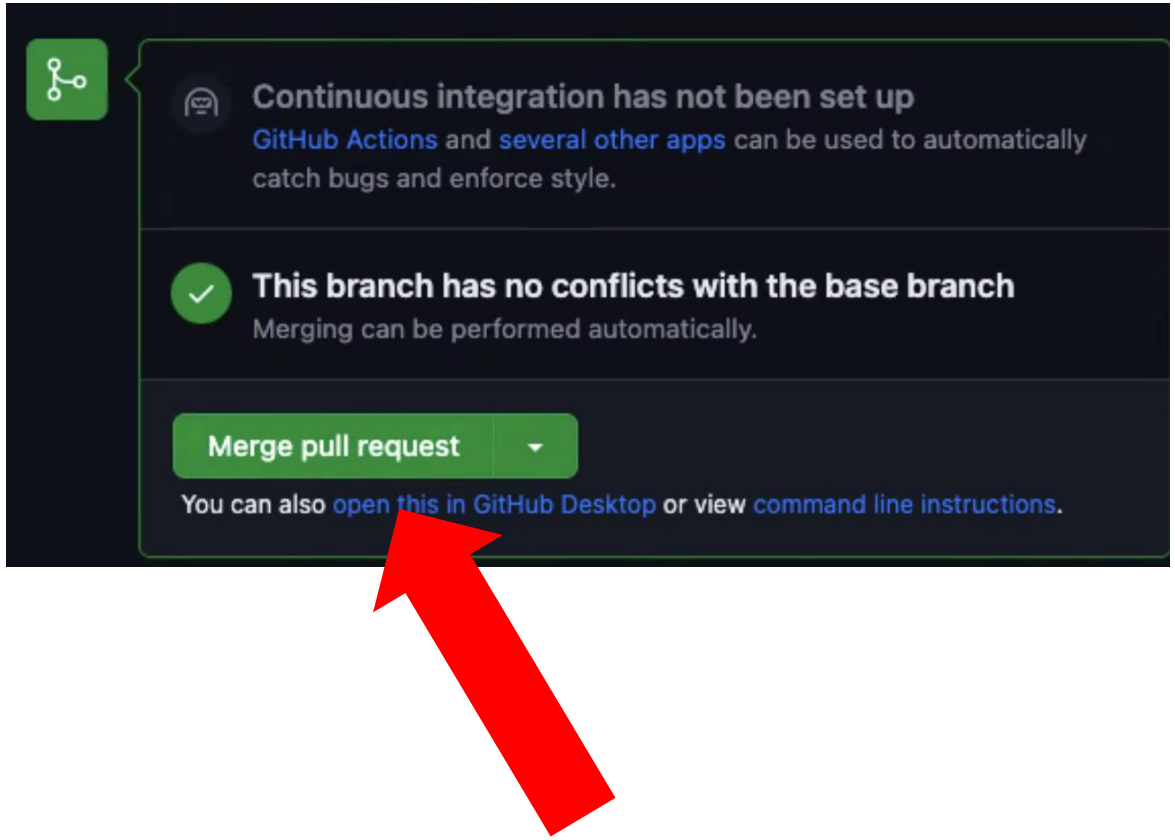


STEP TWO (b): RESOLVE any conflicts

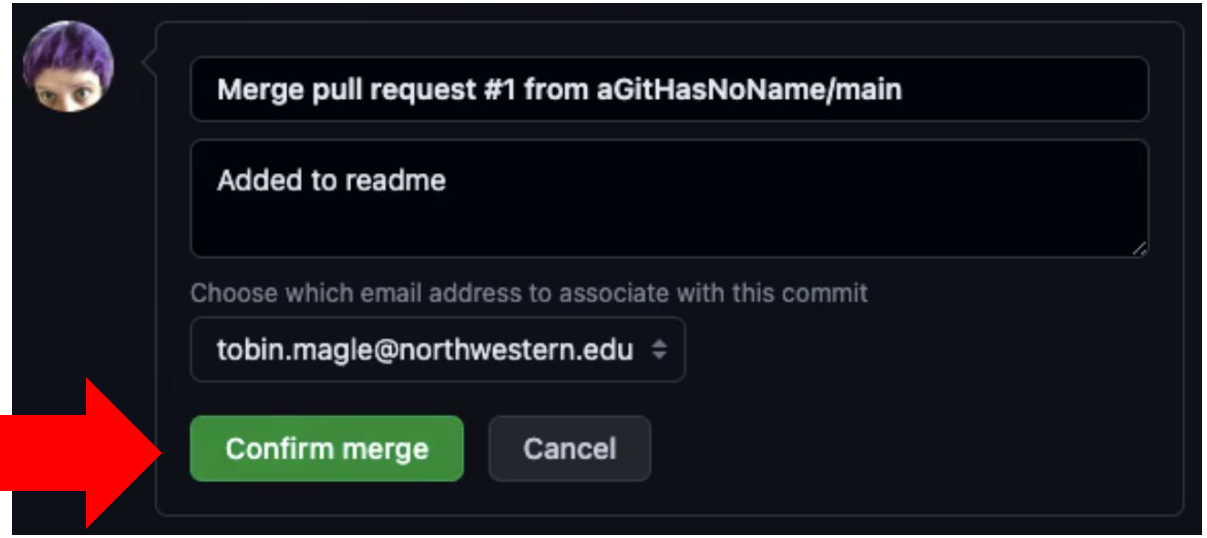


Click the warning box “I understand, update main”.

STEP TWO (b): RESOLVE any conflicts



Click on “Merge pull request”, then “Confirm merge”



Other ways to collaborate

OPEN AN **ISSUE**

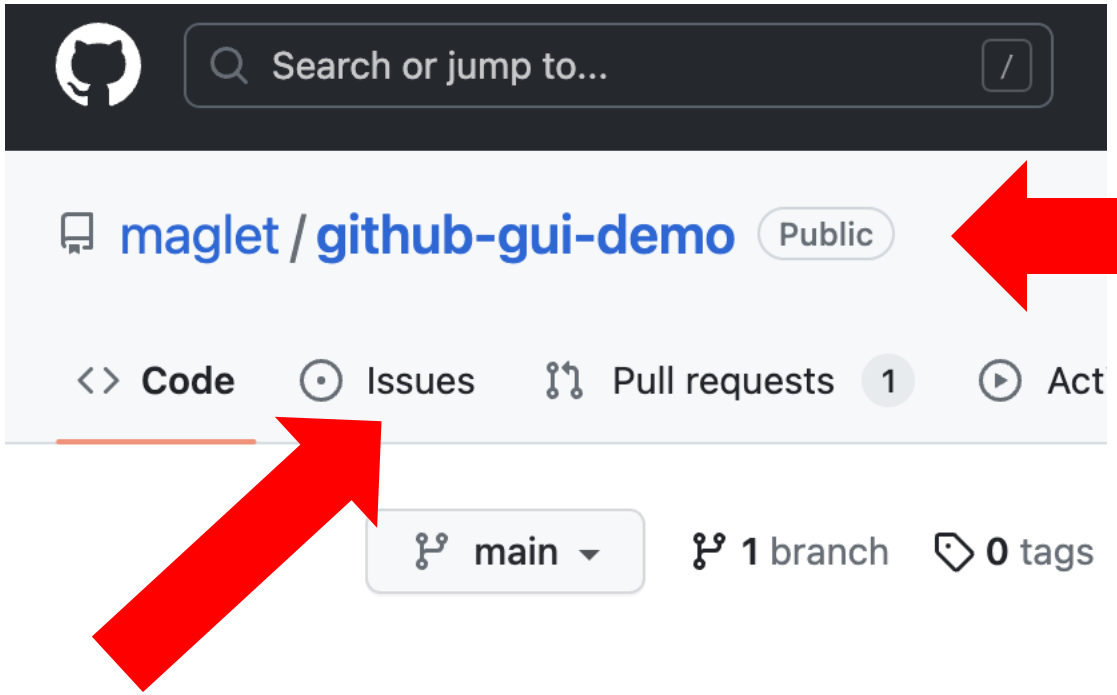
Issues are a great way to

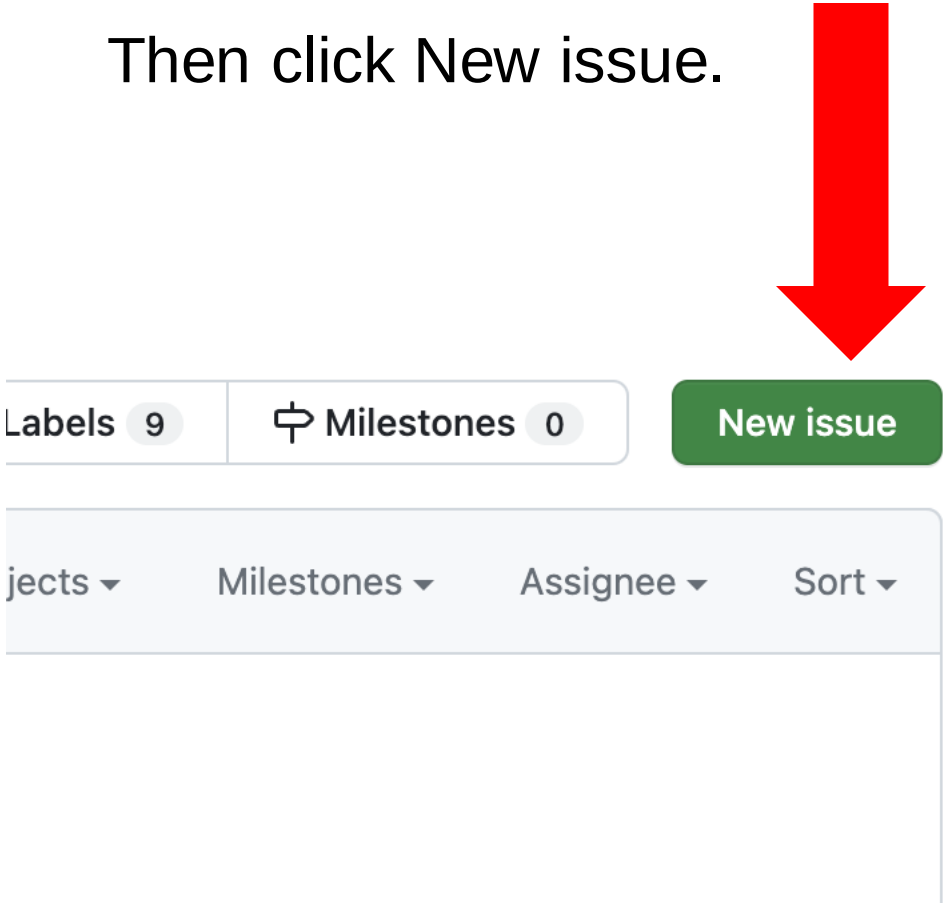
- track what still needs to be done on a project
- report bugs that need to get fixed
- have discussions about the what, why, how, and who of your code and documentation

OPEN AN **ISSUE**

On your friend's repo, click Issues.

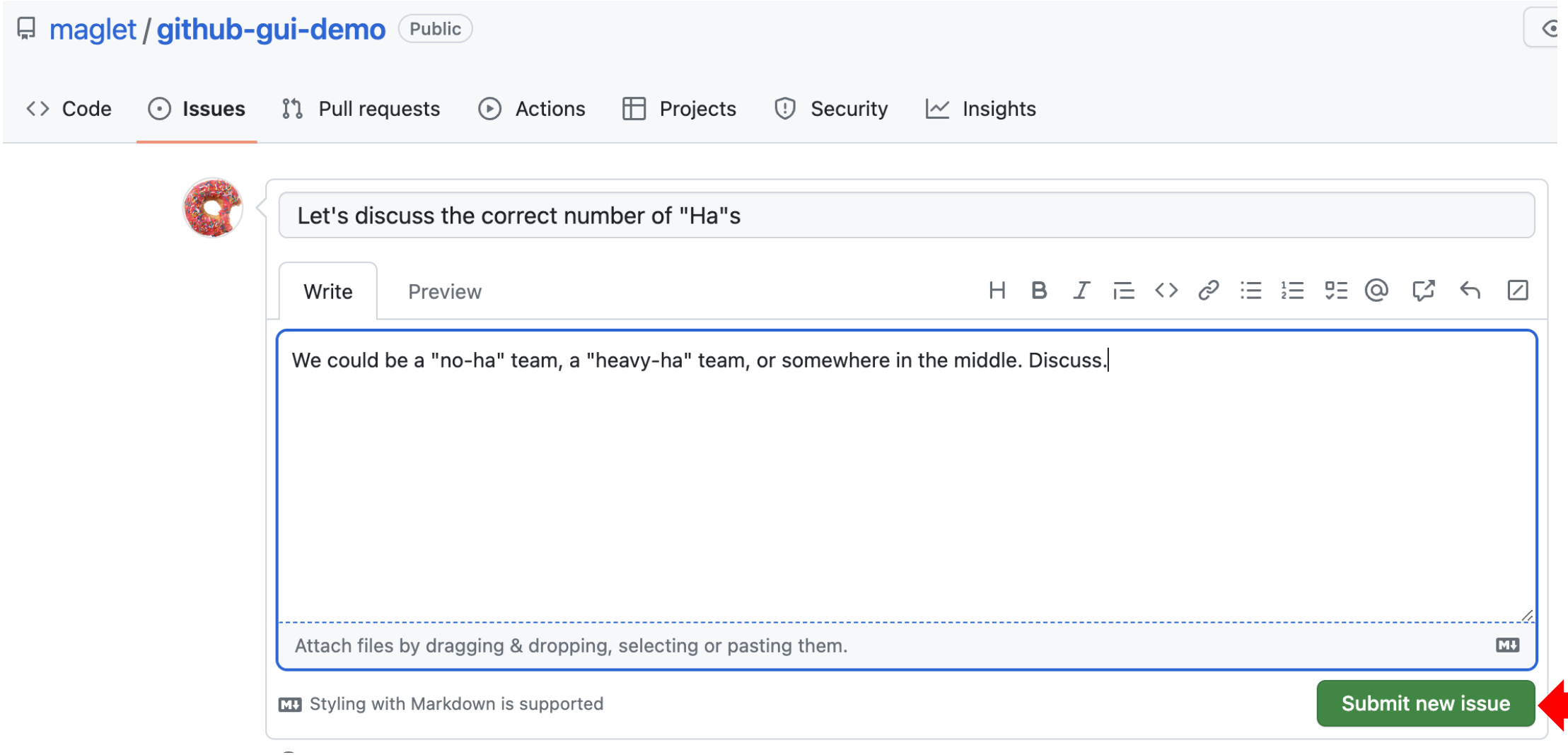
Then click New issue.





OPEN AN **ISSUE**

Submitting an issue will start a discussion thread.




git from the command line

git from the command line

You can also do your commits, pushes, and pulls from the command line (using git commands instead of point-and-click).

You will need a Linux or bash terminal. Mac computers come with a Linux terminal, but PC users will need to install one. We recommend WSL2 or git-bash.

git from the command line

Suggestions for learning git:

- We will teach git this Winter or Spring.
  - Subscribe to the Research Computing Services listserv to get the most up-to-date workshop announcements.
- Follow a self-guided tutorial online. We recommend https://carpentries-incubator.github.io/git-novice-branch-pr/

git from within RStudio

You can make commits and push changes to GitHub from within RStudio. Takes a little bit of know-how to get it linked.

Workshop on Using Git from RStudio: Next Tuesday, October 17th 12pm-1pm on Zoom:

https://planitpurple.northwestern.edu/event/604298