

Quick guide to



git

Flash talk

24.10.2024

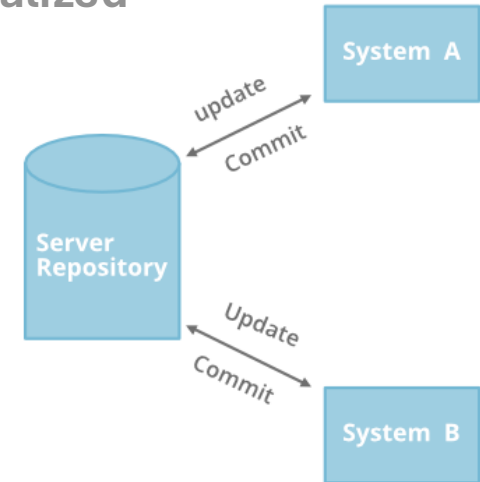
What is Git?

- **distributed** version-control system
- command line
- usually pre-installed (Mac / Linux)

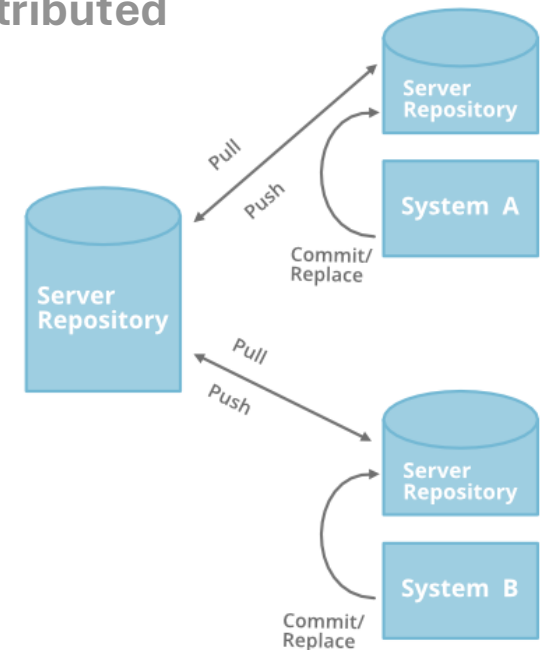
What is GitHub?

- online application that provides:
 - visual interface for `git`
 - free cloud storage for projects' code and files

Centralized



Distributed



Why is GitHub useful?

Version control

- No more *script*, *script_v2*, *script_final*, etc...
- Can go back to any previous versions

Collaboration

- Integrates edits from multiple users
- Keeps record of who did what and when

Documentation

- All the scripts and files of your project are in a single place
- README.md files and Wiki pages

Reproducibility

- You and other researchers can recreate and / or modify any script you created

Visibility

- Showcase your work to others
- Promote your lab's work and facilitate the use of your pipelines

RLAlab

Type to search

+

Overview

Repositories 2

Projects

Packages

Teams 3

People 5

Insights

Settings

RLAlab

RLAlab - Synthetic Biology for Metabolic Engineering

1 follower

United Kingdom

<https://www.rlalab.org>

Unfollow

We think you're gonna like it here.

We've suggested some tasks here in your organization's overview to help you get started.

Invite your people

Invite your first member

Find people by their GitHub username or email address.

Customize members' permissions

Set everyone's base permissions for your code.

View as: Public

You are viewing the README and pinned repositories as a public user.

You can [create a README file](#) or [pin repositories](#) visible to anyone.

You can [hide the tasks we've suggested](#) on this page and bring them back later.

Discussions

Set up discussions to engage with your community!

[Turn on discussions](#)

<https://github.com/RLAlab>

RLA**lab**

Type / to search

+ ▾

Overview

Repositories 2

Projects

Packages

Teams 3

People 5

Insights

Settings

Repositories

All

Public

Private

Sources

Forks

Archived

Templates

All

New repository

Search repositories

2 repositories

↕ Last pushed ▾

☰ ☰

rlalab

Public

source code for RLA**lab** website

● TeX · 📄 MIT License · 👤 0 · ☆ 0 · 🔄 0 · 📡 0 · Updated 2 days ago

Yali_GEM

Public

HTML · 👤 0 · ☆ 0 · 🔄 0 · 📡 0 · Updated on Aug 28

RLA lab

Type to search

Overview

Repositories2

Projects

Packages

Teams3

People5

Insights

Settings

Organization permissions

Members5

Outside collaborators

Pending collaborators

Invitations

Failed invitations

Security Managers

Find a member...

Export

Invite member

Members

2FA

Membership

drsanchis

2FA

Private

Owner

1 team

0 roles

Eliza Atkinson

2FA

Private

Owner

1 team

0 roles

Giorgia Del Missier

2FA

Private

Owner

1 team

0 roles

LCoppens

2FA

Private

Owner

0 teams

0 roles

SAYANTAN

2FA

Private

Member

1 team

0 roles

Find a team...

New team

Select all

Visibility

Members

Computational-team

2 members

0 roles

0 teams

Saccharomyces-team

1 member

0 roles

0 teams

Yarrowia-team

1 member

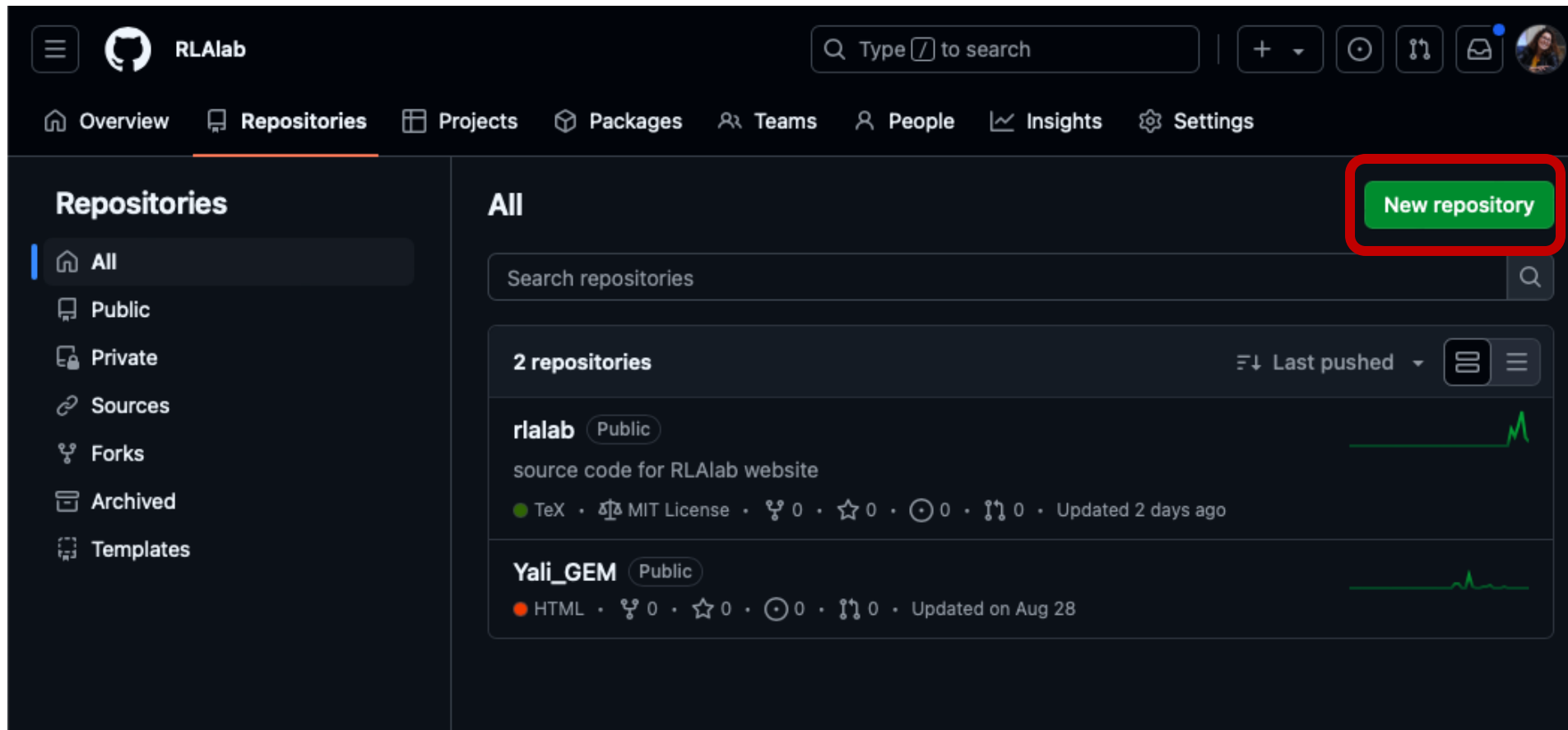
0 roles

0 teams

10 most common **git** terms

1. Repository (repo)

a project that contains all your files, history, ...



10 most common

1. Repository (repo) a project that contain

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 (*).

Repository template

No template ▾

Start your repository with a template repository's contents.

Owner *

RLA lab ▾

Repository name *

git_guide

git_guide is available.

Great repository names are short and memorable. Need inspiration? How about [ubiquitous-spoon](#) ?

Description (optional)

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.

Initialize this repository with:

☒ Add a README file

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

Add .gitignore

.gitignore template: None ▾

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: None ▾

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 RLA lab's [settings](#).

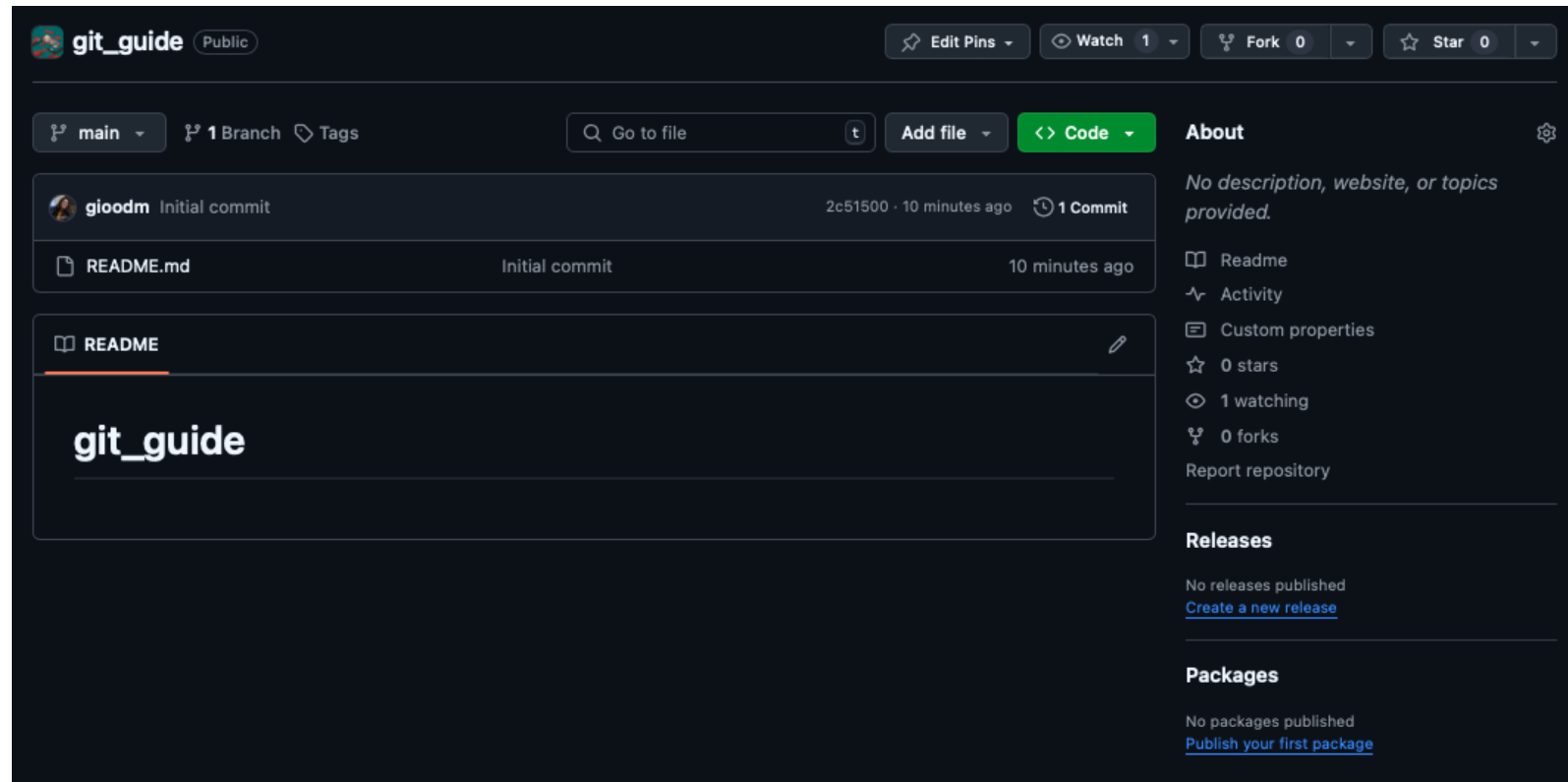
Ⓒ You are creating a public repository in the RLA lab organization.

Create repository

10 most common **git** terms

1. Repository (repo)

a project that contains all your files, history, ...

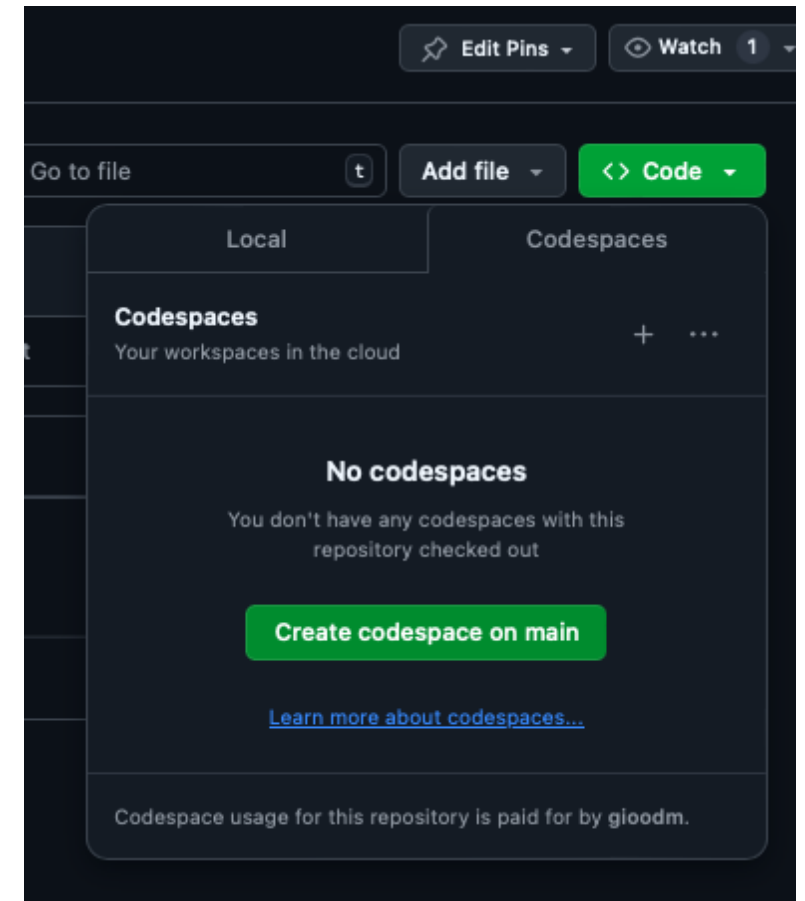
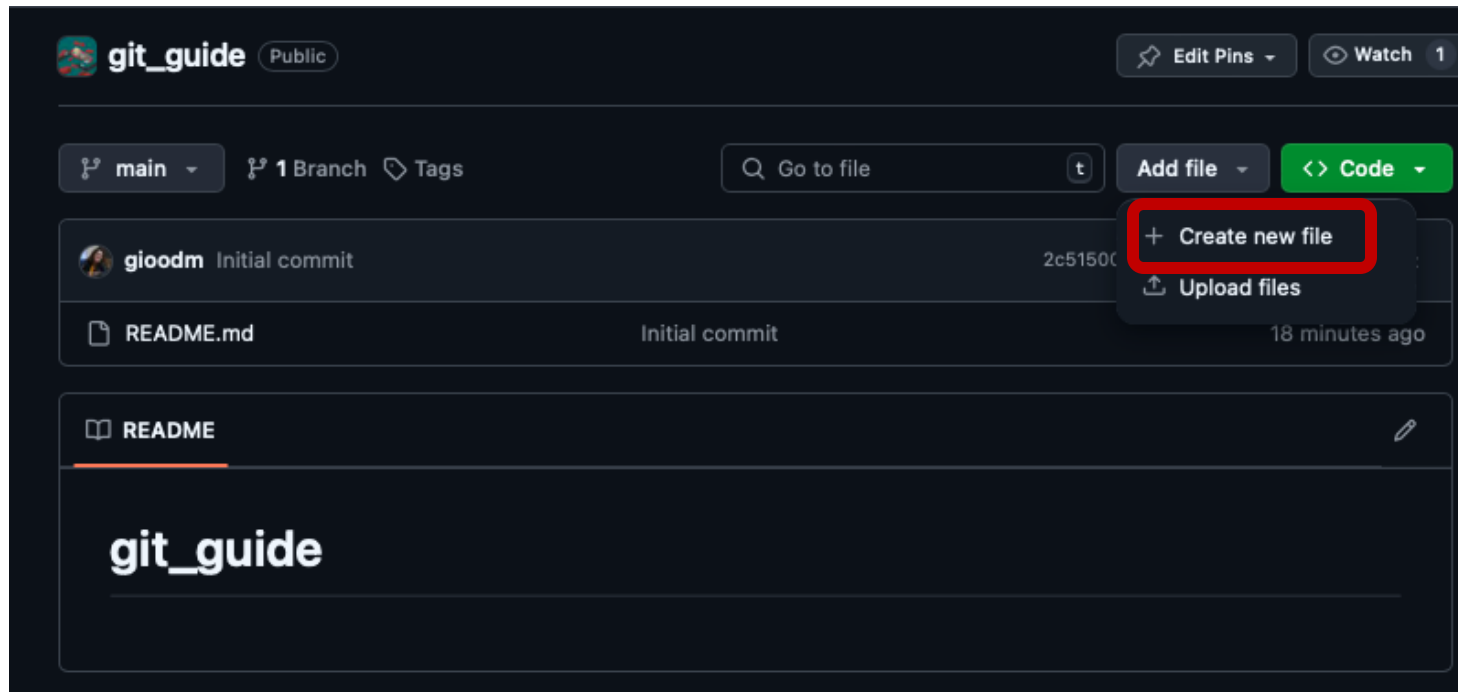


```
~% mkdir git_guide
~% cd git_guide
~% git init
```

10 most common **git** terms

2. commit

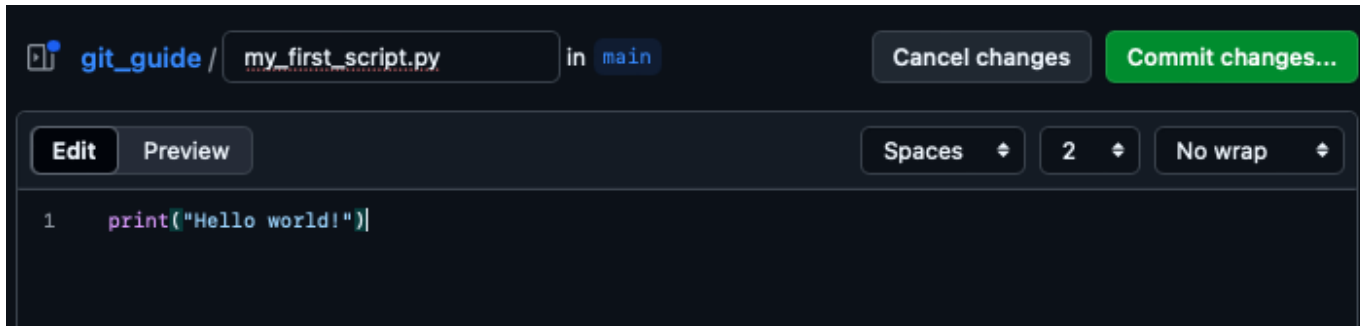
a snapshot of changes in the repository



10 most common **git** terms

2. commit

a snapshot of changes in the repository

A screenshot of a 'Commit changes' dialog box. It has a title bar with a close button. The 'Commit message' field contains the text 'Create my_first_script.py|'. Below it is an 'Extended description' field with the placeholder text 'Add an optional extended description..'. At the bottom, there are two radio button options: 'Commit directly to the main branch' (which is selected) and 'Create a new branch for this commit and start a pull request'. A red dashed box highlights the second option and the link 'Learn more about pull requests' below it. At the bottom right are 'Cancel' and 'Commit changes' buttons.

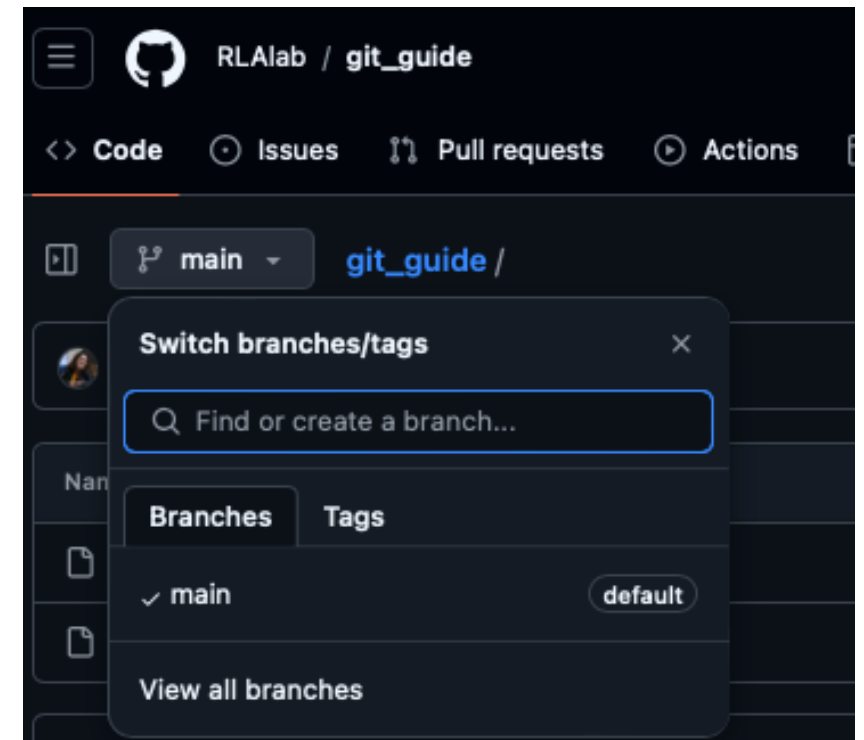
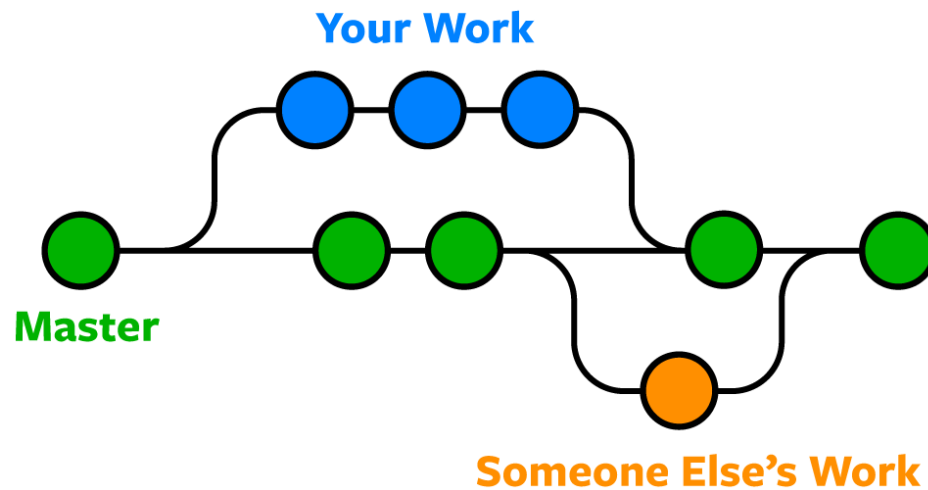
~% **git commit -m** "message"

10 most common **git** terms

3. branch

a separate line of development

`master / main` is the default branch



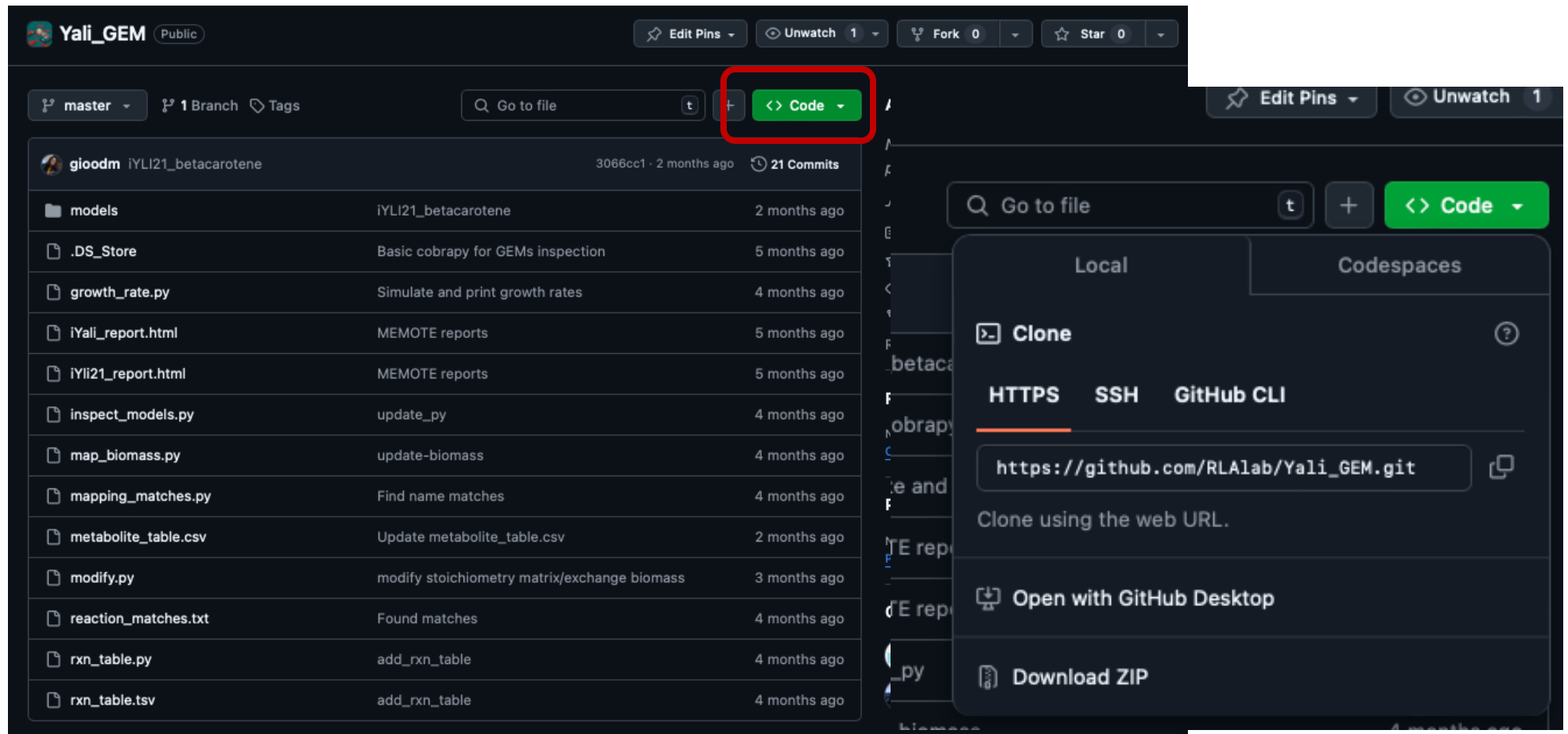
```
~% git branch # to list branches
~% git checkout -b branch-name # to create a new branch and switch to it

~% git branch branch-name # creates a branch
~% git checkout branch-name # switch to the new branch
```

10 most common **git** terms

4. clone

a local copy of a remote repository

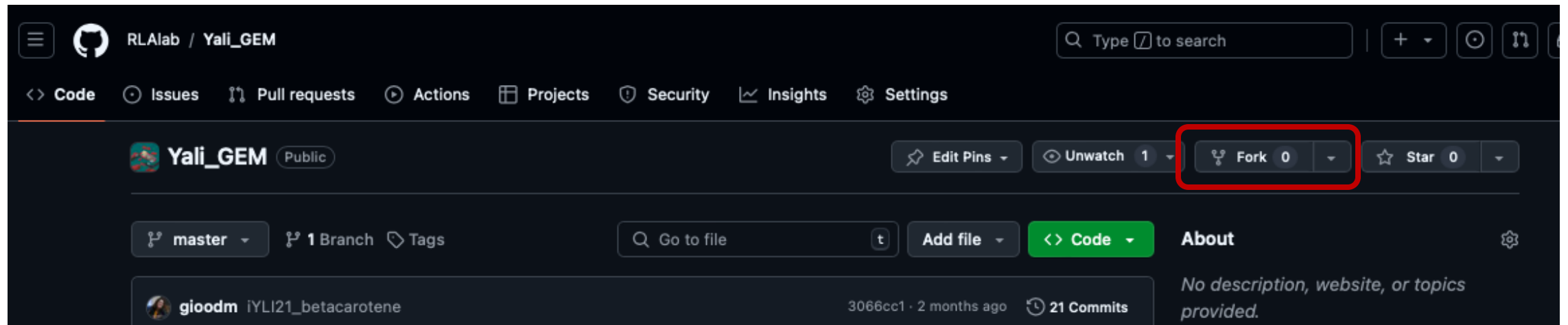


~% git clone <url>

10 most common **git** terms

5. fork

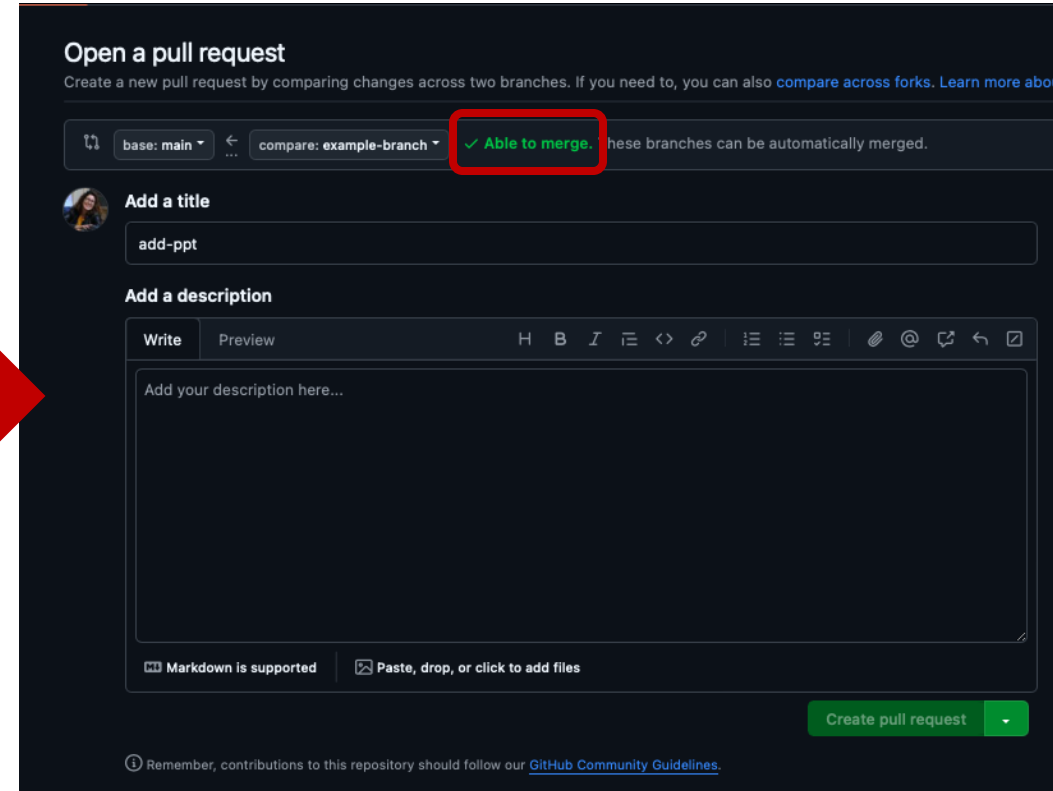
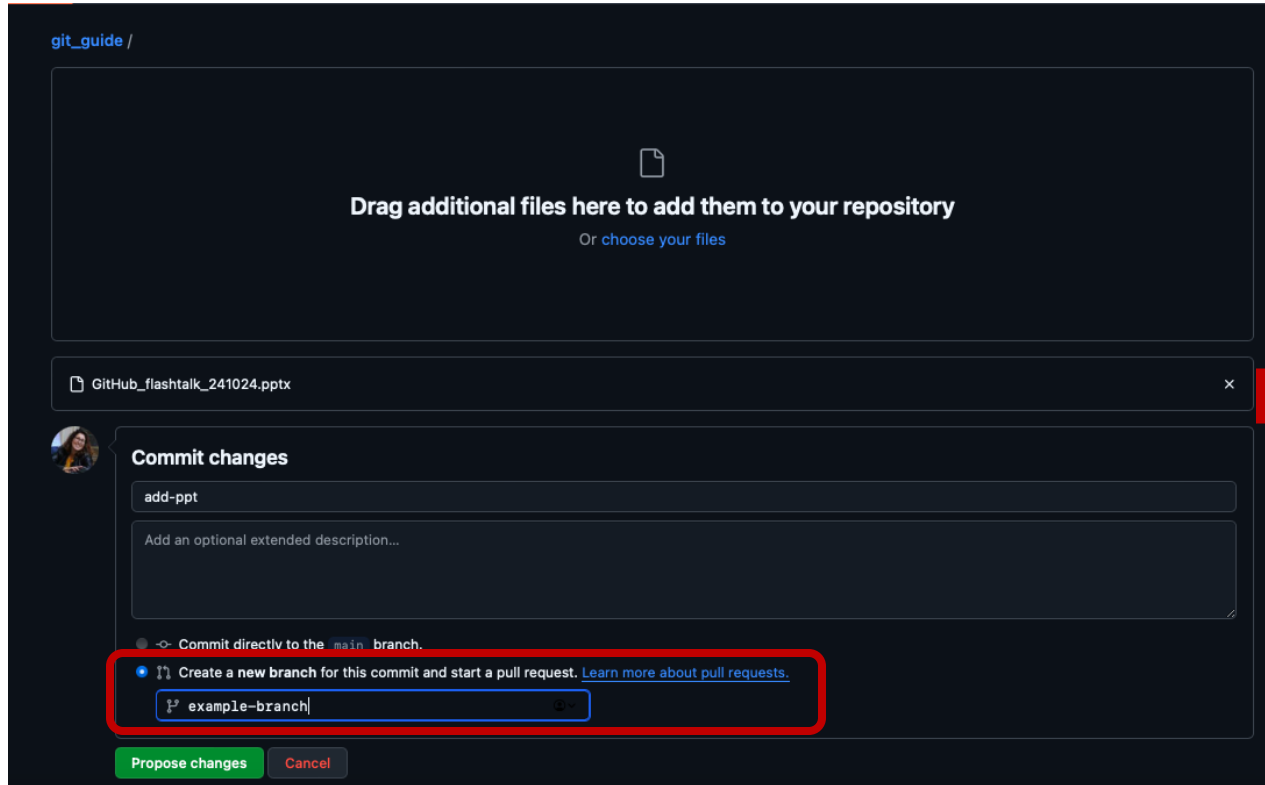
a copy of someone else's repository on your account



10 most common **git** terms

6. pull request (PR)

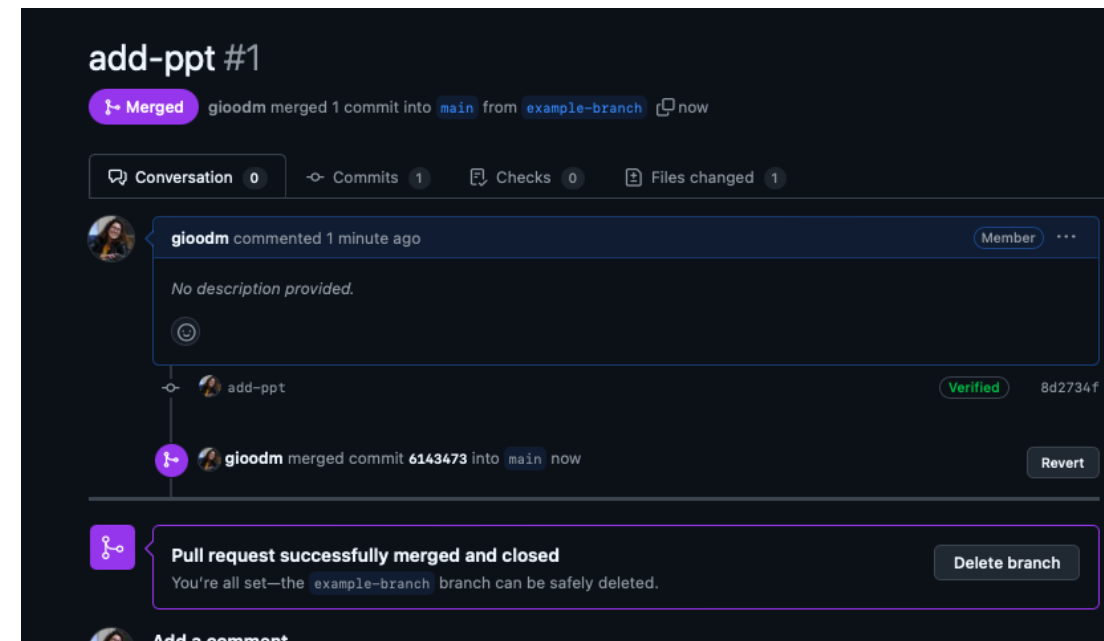
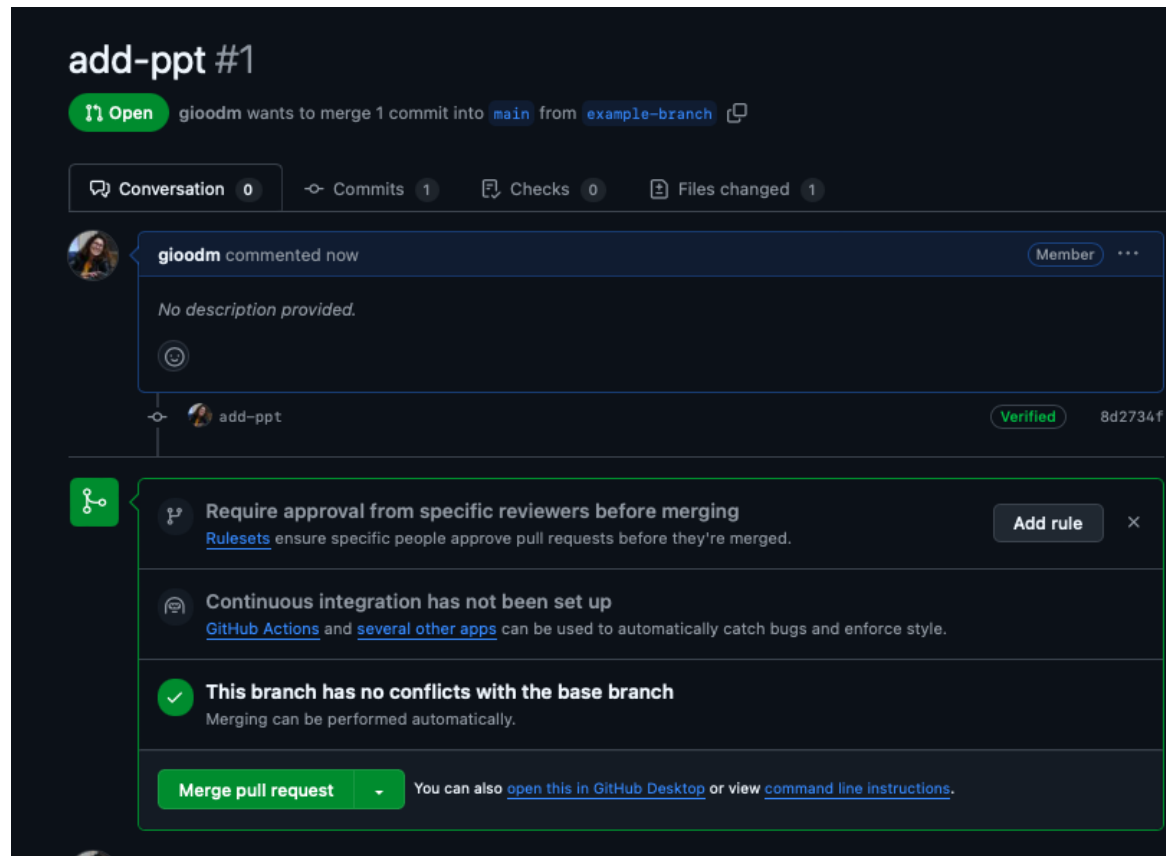
a request to merge changes from one branch into another
(usually from a feature branch into the main branch)



10 most common **git** terms

7. merge

combining changes from one branch into another



10 most common **git** terms

8. pull

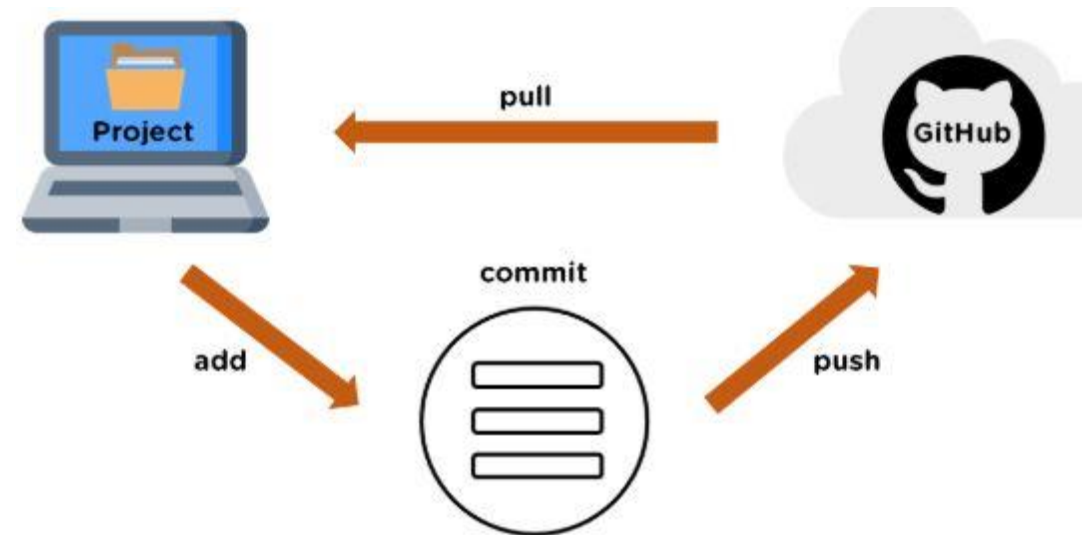
fetch and integrate changes from the remote repository into your local copy

9. staging (area)

the area where changes are prepared before committing

10. push

sending local commits to the remote repository



```
~% git pull
~% git add <file>
~% git push origin branch-name
```

Cheat sheet

Create a Repository

From scratch -- Create a new local repository

```
$ git init [project name]
```

Download from an existing repository

```
$ git clone my_url
```

Observe your Repository

List new or modified files not yet committed

```
$ git status
```

Show the changes to files not yet staged

```
$ git diff
```

Show the changes to staged files

```
$ git diff --cached
```

Show all staged and unstaged file changes

```
$ git diff HEAD
```

Show the changes between two commit ids

```
$ git diff commit1 commit2
```

List the change dates and authors for a file

```
$ git blame [file]
```

Show the file changes for a commit id and/or file

```
$ git show [commit]:[file]
```

Show full change history

```
$ git log
```

Show change history for file/directory including diffs

```
$ git log -p [file/directory]
```

Working with Branches

List all local branches

```
$ git branch
```

List all branches, local and remote

```
$ git branch -av
```

Switch to a branch, my_branch, and update working directory

```
$ git checkout my_branch
```

Create a new branch called new_branch

```
$ git branch new_branch
```

Delete the branch called my_branch

```
$ git branch -d my_branch
```

Merge branch_a into branch_b

```
$ git checkout branch_b
```

```
$ git merge branch_a
```

Tag the current commit

```
$ git tag my_tag
```

Make a change

Stages the file, ready for commit

```
$ git add [file]
```

Stage all changed files, ready for commit

```
$ git add .
```

Commit all staged files to versioned history

```
$ git commit -m "commit message"
```

Commit all your tracked files to versioned history

```
$ git commit -am "commit message"
```

Unstages file, keeping the file changes

```
$ git reset [file]
```

Revert everything to the last commit

```
$ git reset --hard
```

Synchronize

Get the latest changes from origin (no merge)

```
$ git fetch
```

Fetch the latest changes from origin and merge

```
$ git pull
```

Fetch the latest changes from origin and rebase

```
$ git pull --rebase
```

Push local changes to the origin

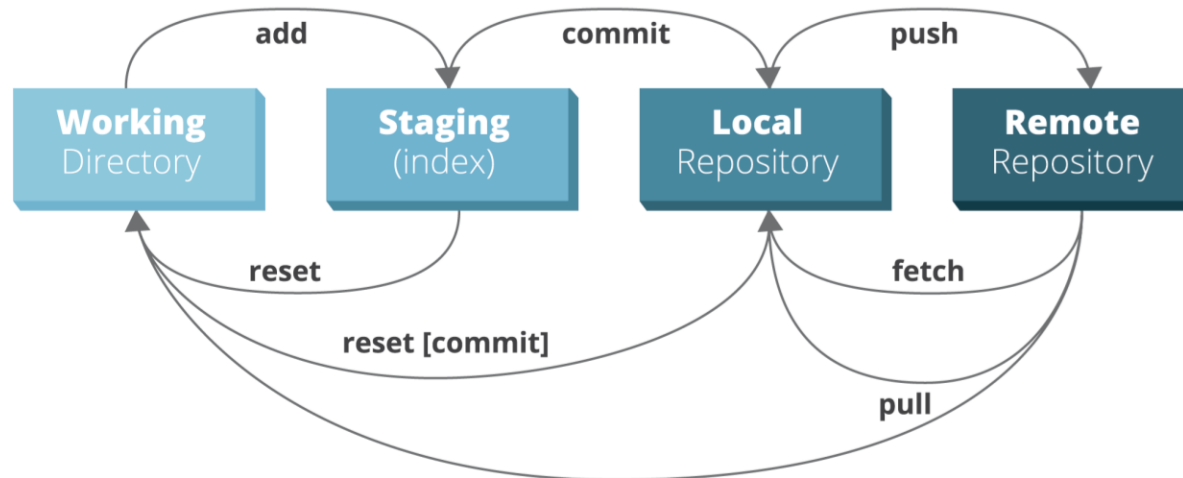
```
$ git push
```

Finally!

When in doubt, use git help

```
$ git command --help
```

Or visit <https://training.github.com/> for official GitHub training.



- <https://www.dataschool.io/git-quick-reference-for-beginners/>
- <https://www.dataschool.io/simple-guide-to-forks-in-github-and-git/>
- <https://www.dataschool.io/how-to-contribute-on-github/>
- <https://www.geeksforgeeks.org/ultimate-guide-git-github/>
- <https://medium.com/@sachinsoni600517/complete-tutorial-of-git-and-github-for-basic-to-advanced-1dd34d12b90b>