Git collaboration - a practical session

Before we start

Requirements

These exercises are designed to be done in pairs.

Each will need

- a laptop with git installed
  - "Installing Git"
- a GitHub account with SSH keys installed
  - "Getting started with your GitHub account"
  - "Adding a new SSH key to your GitHub account"

You will be asked to open a terminal (also called a "shell") to use git commands. - On Windows, you can use the "Windows Console" (`cmd.exe`), or on more recent version versions of Windows either "PowerShell" or the built-in "Terminal" application. - On MacOS, you can use the built-in "Terminal" application. - On Linux, use the terminal emulator of your choice.

Configure git If you've never used git before, you will need to configure it on your laptop. Use the following commands (replace with your own information).

```
git config --global user.name "First name Last name"
git config --global user.email firstname.lastname@example.com
```

You should also make sure that the default name for the main branch is "main".

```
git config --global init.defaultBranch main
```

By default, the editor used in the terminal could be "vim", which can be a bit rough for beginners.

On Windows, change it to "notepad.exe"

```
git config --global core.editor notepad
```

On Linux and MacOS, change it to "nano" (read about nano).

```
git config --global core.editor "nano"
```

A few tips

- Use `git status` a lot: before and after every `git` command!
- You can print the entire graph with `git log --all --graph --oneline`
- Don't hesitate to look for answers on Google!

Warnings

- Be very careful whenever you use `-f` or `--force` in any command! Any forced operation might delete data permanently.

Links and references

- Official git documentation: https://git-scm.com/docs
- Atlassian's git tutorial: https://www.atlassian.com/git/tutorials/setting-up-a-repository

Agenda

The goal of this practical session is to demonstrate how git enables collaborative work by allowing you to share your contributions and to manage conflicts.

You will learn

- how to make commits

- how to fetch commits made by others
- how to create a branch and switch between branches
- how to merge your changes with others
- how to share your commits

Collaboration using git

Setting things up

Collaboration requires a remote repository. We'll be using GitHub to get one for free.

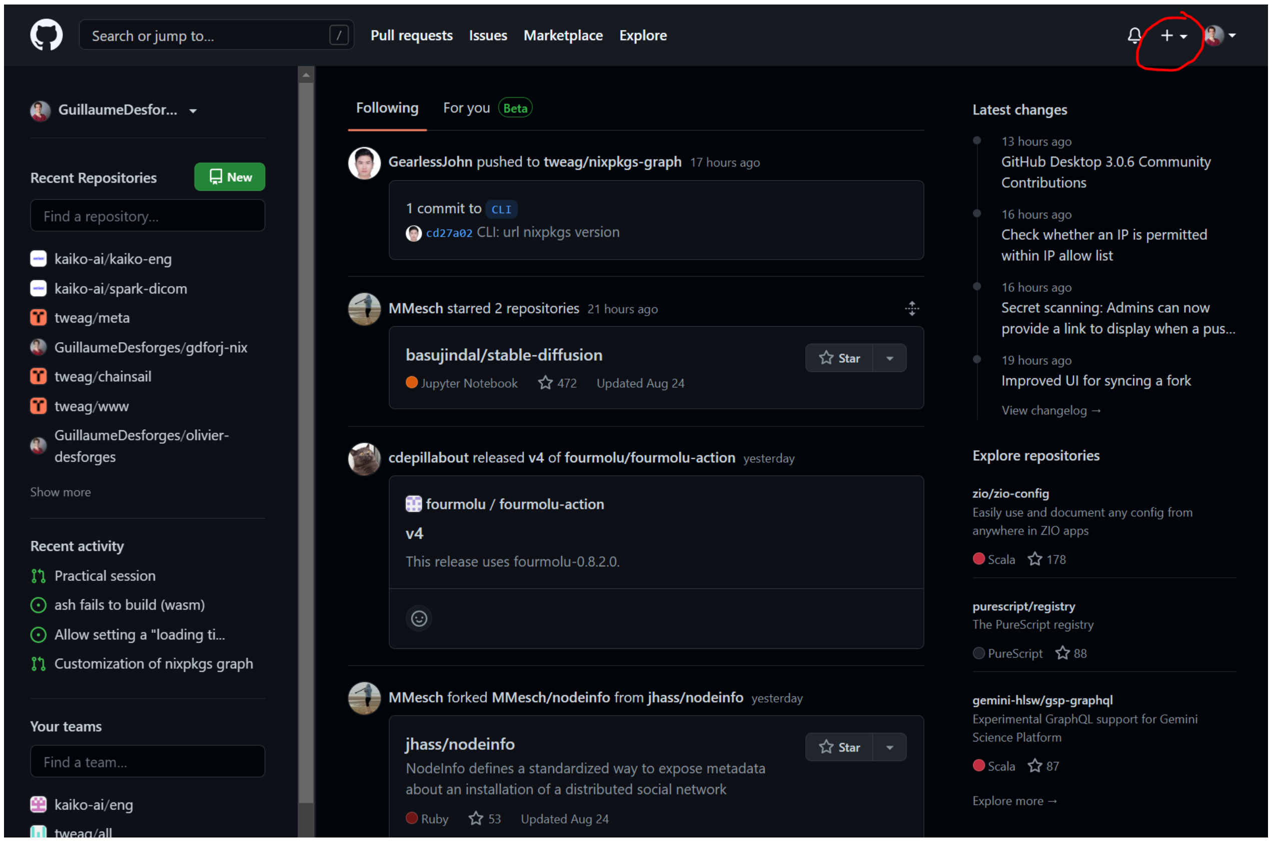
https://github.com/

Create a GitHub project This has to be done by only one of the two partners, as it will be shared.

Check that you are logged in to GitHub!

1. Create a repository

- click on "+" in the top right corner



GearlessJohn pushed to tweag/nixpkgs-graph 17 hours ago
1 commit to CLI
cd27a02 CLI: url nixpkgs version

MMesch starred 2 repositories 21 hours ago
basujindal/stable-diffusion
Jupyter Notebook 472 Updated Aug 24

cdepillabout released v4 of fourmolu/fourmolu-action yesterday
fourmolu / fourmolu-action
v4
This release uses fourmolu-0.8.2.0.

MMesch forked MMesch/nodeinfo from jhass/nodeinfo yesterday
jhass/nodeinfo
NodeInfo defines a standardized way to expose metadata about an installation of a distributed social network
Ruby 53 Updated Aug 24

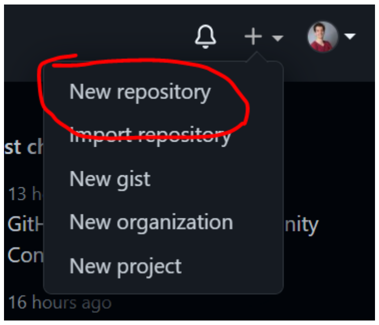
13 hours ago
GitHub Desktop 3.0.6 Community Contributions
16 hours ago
Check whether an IP is permitted within IP allow list
16 hours ago
Secret scanning: Admins can now provide a link to display when a pus...
19 hours ago
Improved UI for syncing a fork

zio/zio-config
Easily use and document any config from anywhere in ZIO apps
Scala 178

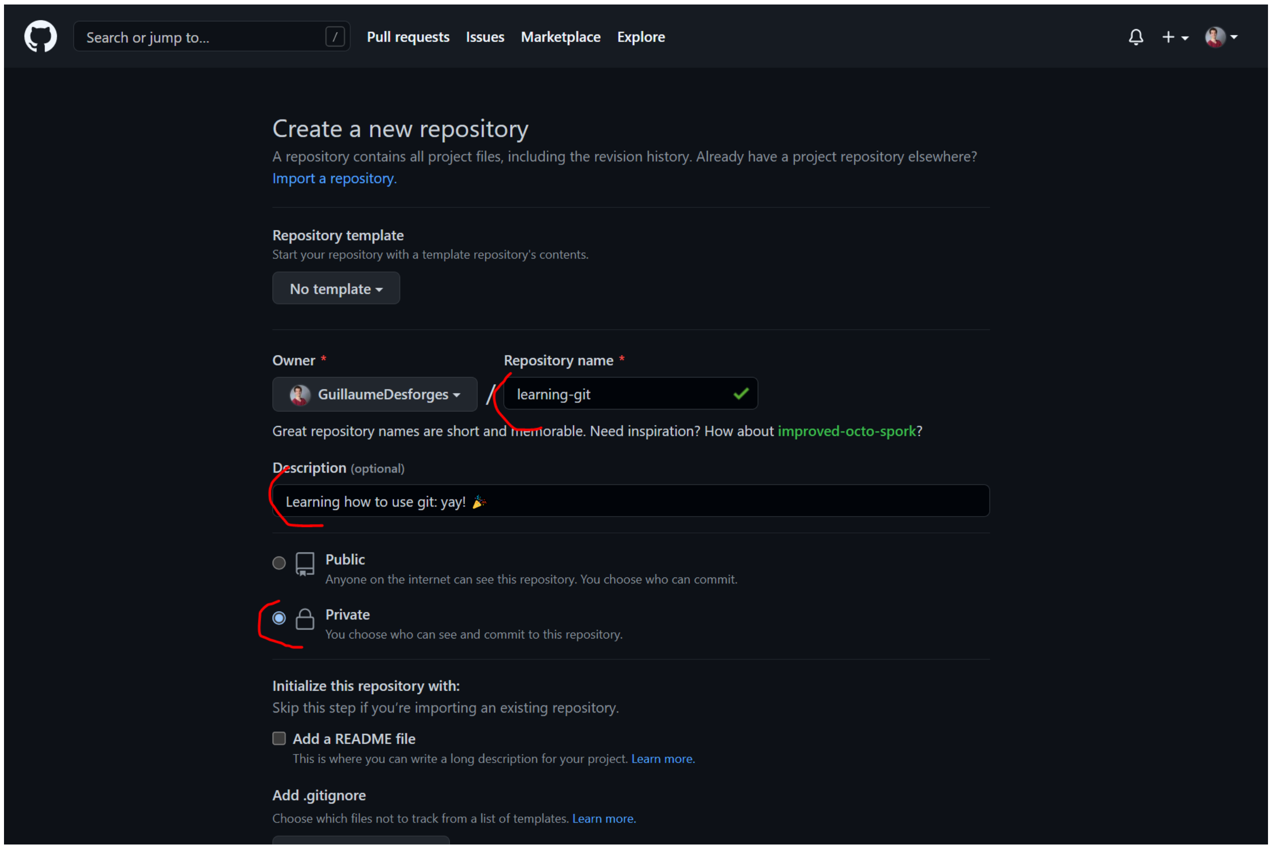
purescript/registry
The PureScript registry
PureScript 88

gemini-hlsw/gsp-graphql
Experimental GraphQL support for Gemini Science Platform
Scala 87

- click on "New repository"



- you can name your repository however you want > Usually, we use lowercase characters (`a`, `b`, `c`, ...) spearated by hyphens `-` instead of spaces. > Example: if your project is name “My awesome Git project”, name your repository `my-awesome-git-project`.
- write a description > Usually a single sentence.
- select “Private”



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Repository template

Start your repository with a template repository's contents.

No template

Owner * GuillaumeDesforges / Repository name * learning-git ✓

Great repository names are short and memorable. Need inspiration? How about improved-octo-spork?

Description (optional)

Learning how to use git: yay! 🎉

( ) 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:

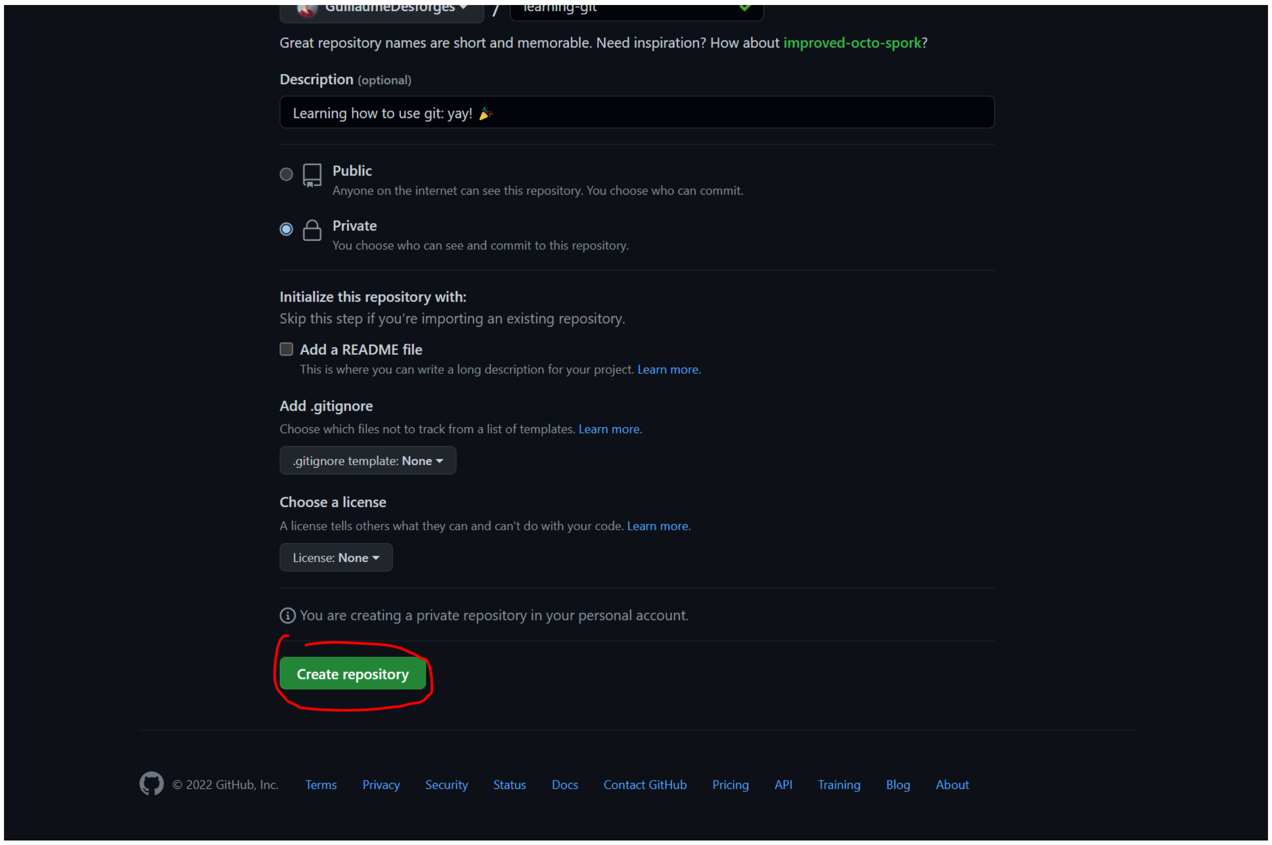
Skip this step if you're importing an existing repository.

☐ Add a README file
This is where you can write a long description for your project. Learn more.

Add .gitignore

Choose which files not to track from a list of templates. Learn more.

- click on the “New repository” button at the end


GuillaumeDesforges / learning-git

Great repository names are short and memorable. Need inspiration? How about improved-octo-spork?

Description (optional)

Learning how to use git: yay! 🎉

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:
Skip this step if you're importing an existing repository.

Add a README file
This is where you can write a long description for your project. Learn more.

Add .gitignore
Choose which files not to track from a list of templates. Learn more.

.gitignore template: None

Choose a license
A license tells others what they can and can't do with your code. Learn more.

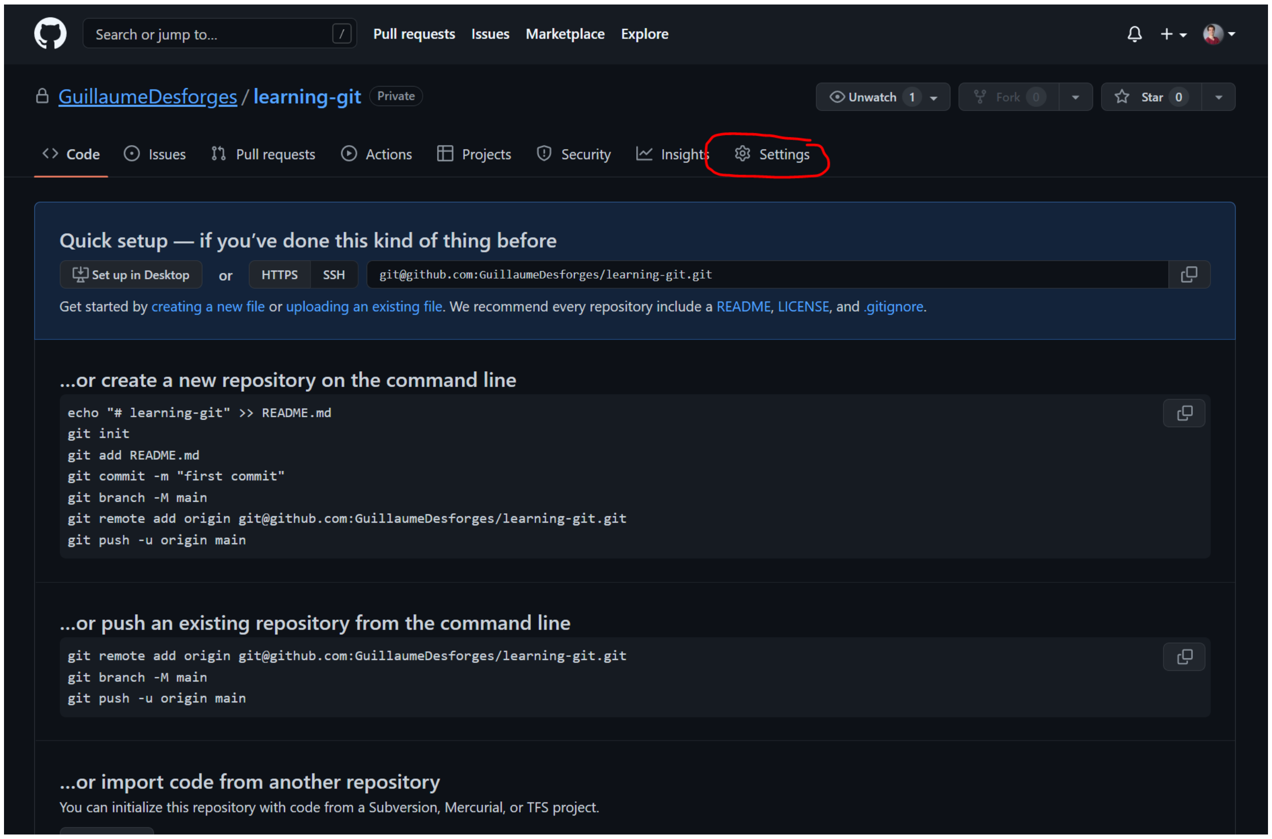
License: None

You are creating a private repository in your personal account.

© 2022 GitHub, Inc. Terms Privacy Security Status Docs Contact GitHub Pricing API Training Blog About

2. Give access to you partner

- go to the web page of your new repository (if you just created it, you should be on it already)
- go to the repository settings ("Settings" tab)



Quick setup — if you've done this kind of thing before

git@github.com:GuillaumeDesforges/learning-git.git

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

```
echo "# learning-git" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin git@github.com:GuillaumeDesforges/learning-git.git
git push -u origin main
```

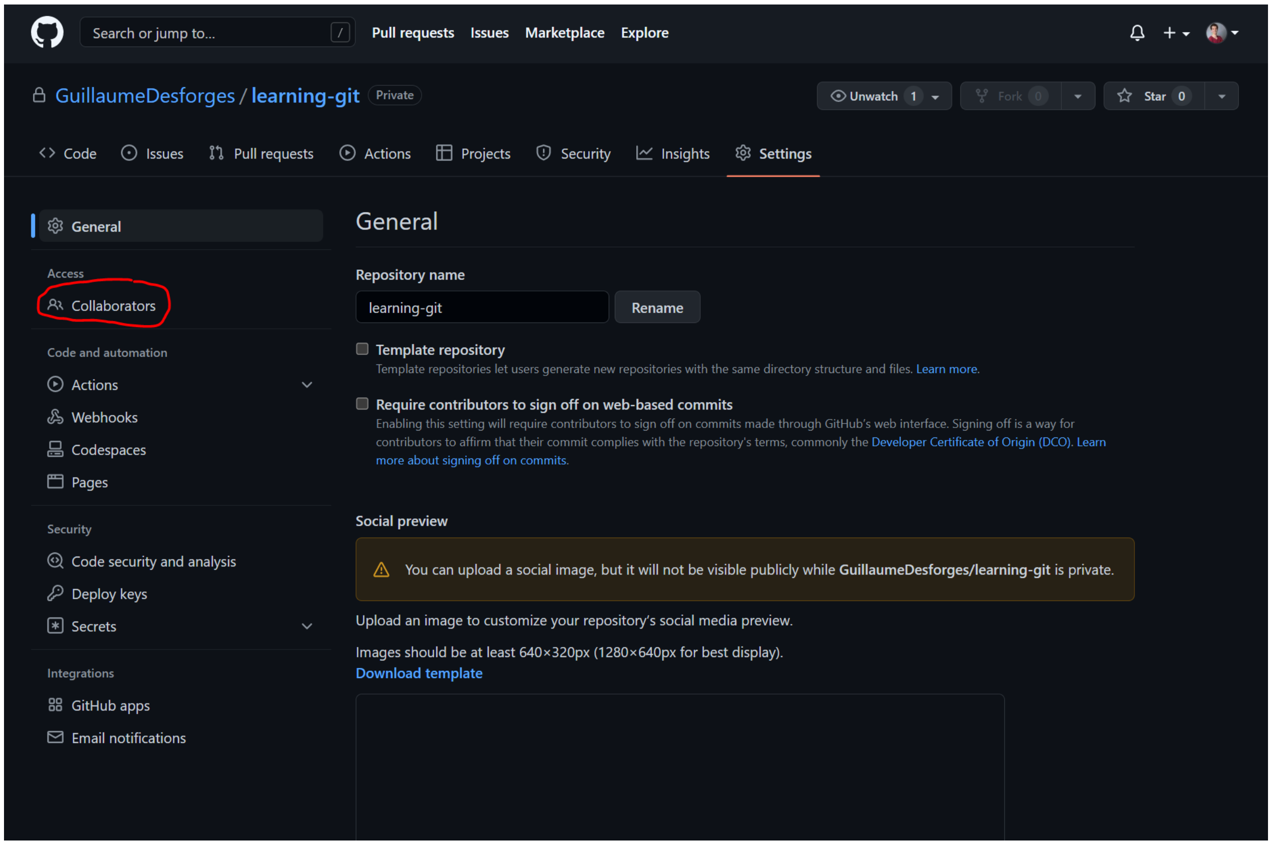
...or push an existing repository from the command line

```
git remote add origin git@github.com:GuillaumeDesforges/learning-git.git
git branch -M main
git push -u origin main
```

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

- go to the “Collaborators” tab (under “Access” group in the left panel)



General

Repository name: learning-git

Template repository
Template repositories let users generate new repositories with the same directory structure and files. Learn more.

Require contributors to sign off on web-based commits
Enabling this setting will require contributors to sign off on commits made through GitHub's web interface. Signing off is a way for contributors to affirm that their commit complies with the repository's terms, commonly the Developer Certificate of Origin (DCO). Learn more about signing off on commits.

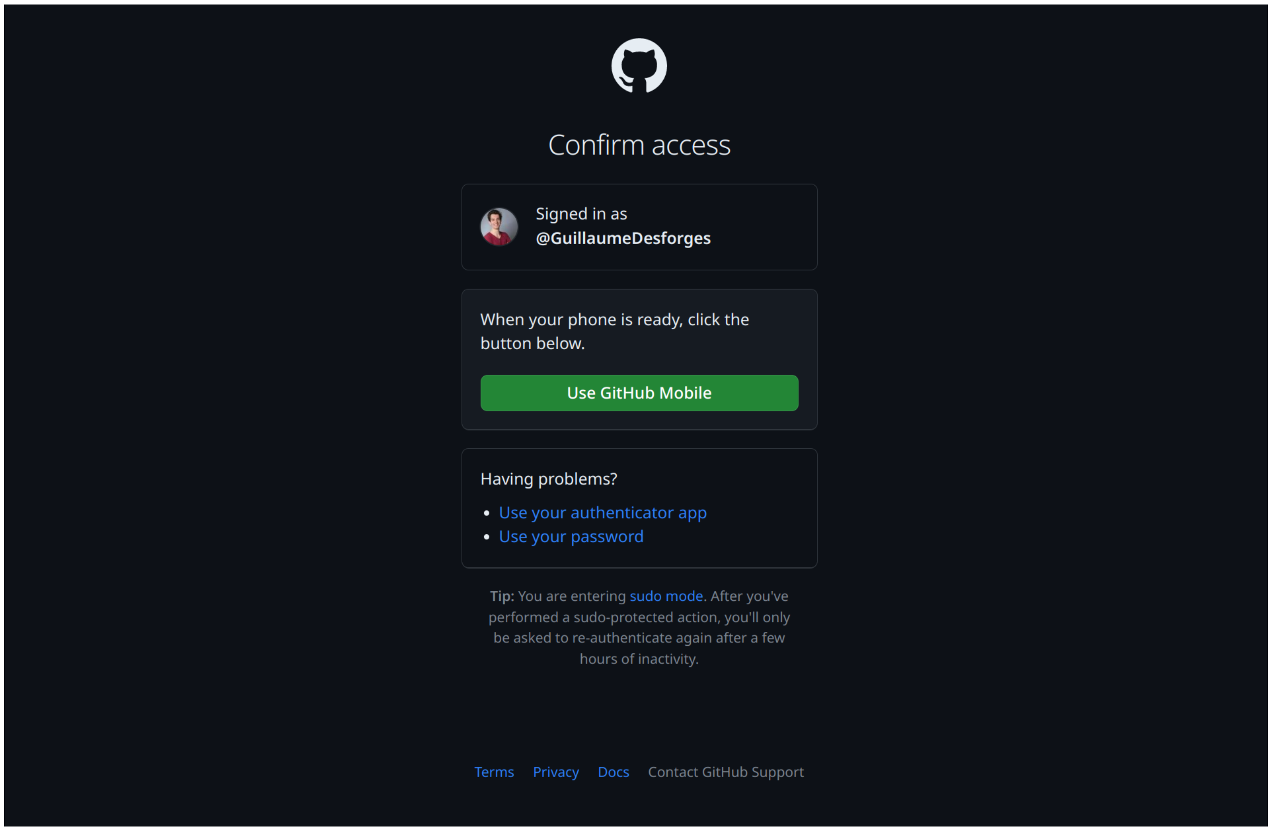
Social preview

You can upload a social image, but it will not be visible publicly while **GuillaumeDesforges/learning-git** is private.

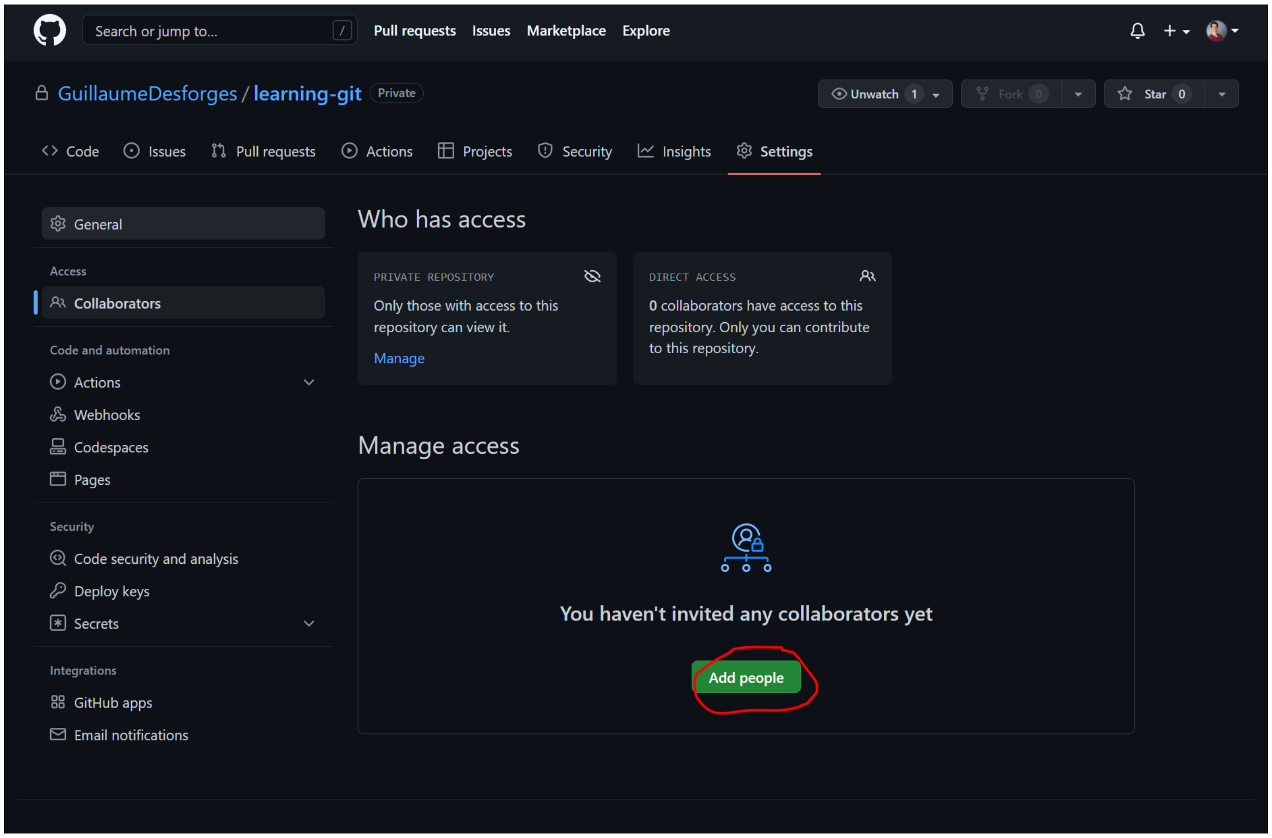
Upload an image to customize your repository's social media preview.

Images should be at least 640×320px (1280×640px for best display).
Download template

you might be required to authenticate via MFA (if not, skip this)


Confirm access
Signed in as
@GuillaumeDesforges
When your phone is ready, click the button below.
Use GitHub Mobile
Having problems?
Use your authenticator app
Use your password
Tip: You are entering sudo mode. After you've performed a sudo-protected action, you'll only be asked to re-authenticate again after a few hours of inactivity.
Terms
Privacy
Docs
Contact GitHub Support

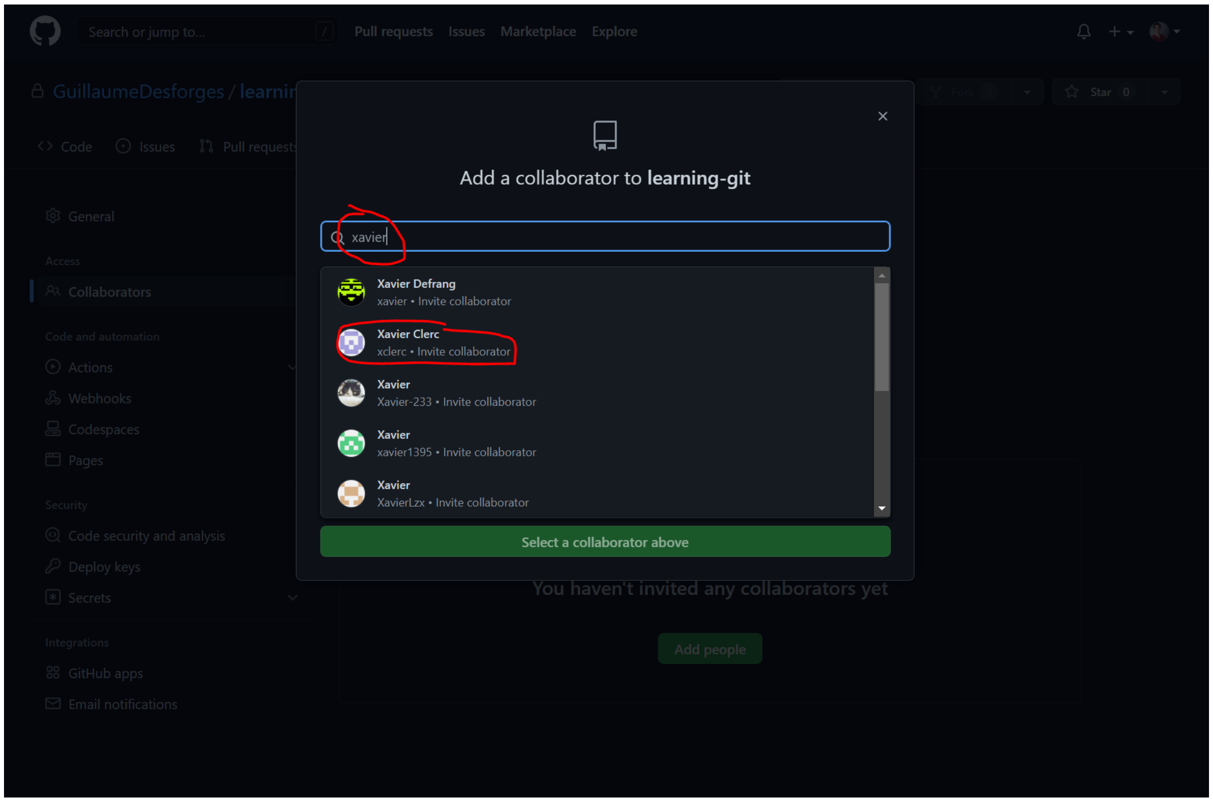
- click on “Add people”



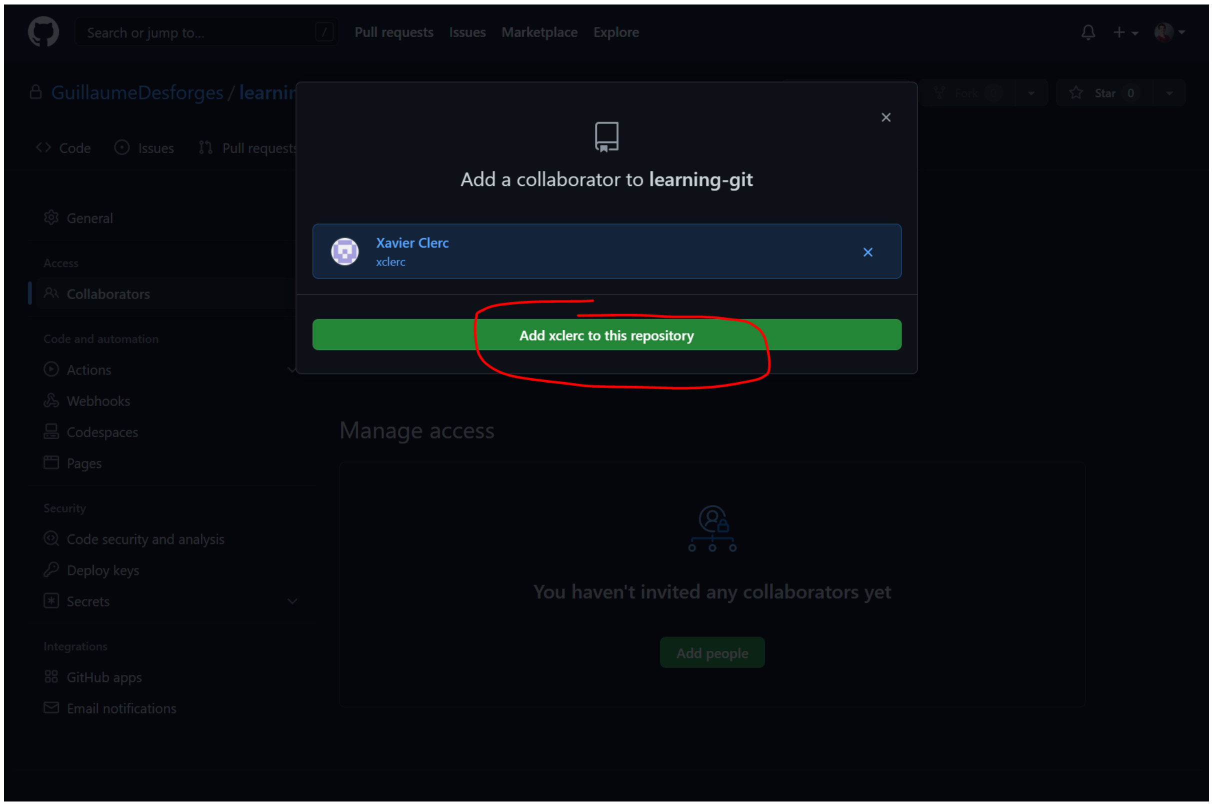
- find and add your partner

– type their GitHub "handle" in the text input

– suggestions should be shown, click on the right one


Search or jump to...
Pull requests
Issues
Marketplace
Explore
GuillaumeDesforges / learnin
Code
Issues
Pull requests
General
Access
Collaborators
Code and automation
Actions
Webhooks
Codespaces
Pages
Security
Code security and analysis
Deploy keys
Secrets
Integrations
GitHub apps
Email notifications
Add a collaborator to learning-git
xavier
Xavier Defrang
xavier • Invite collaborator
Xavier Clerc
xclerc • Invite collaborator
Xavier
Xavier-233 • Invite collaborator
Xavier
xavier1395 • Invite collaborator
Xavier
XavierLzx • Invite collaborator
Select a collaborator above
You haven't invited any collaborators yet
Add people

– when found, click on the button below "Add XXX to this repository"



Download the repository to your laptop On the repository web page in GitHub, as the project is empty for now, you'll find a "Quick setup" banner. Click on the "SSH" button in there, you should see something in the form of:

```
git@github.com:yourHandle/your-repo-name.git
```

Copy this url, and clone it to your laptop, for example:

```
git clone git@github.com:yourHandle/your-repo-name.git
```

where `your-repo-name` is replaced by the name of your own repository.

A folder must have been created with this name (e.g. `your-repo-name`).

Change your working directory to it:

```
cd your-repo-name
```

where `your-repo-name` is replaced by the name of your own repository.

Congratulations! You just "cloned" the repository from the remote (GitHub). Right now this folder is empty, but soon enough we'll fill it with some files.

> If you have enabled "See hidden files" option, notice the `.git` folder.
>
> git stores all the data it needs to manage the repository in this `.git` folder. If you delete it, everything you haven't pushed will be lost forever!

Hands-on

Now, it's time to start for real!

We'll go through several scenarios which you will often encounter.

One person will play the role of "computer 1" and the other the role of "computer 2".

> Though the examples might seem trivial, pay special attention to understand what is happening.

First scenario: collaboration without conflict We'll start with the simplest.

- On computer 1
  - Create a new file in the folder of the repository, with the following Python script
    ```python
    print("Hello world")
    ```
  - Commit this new file to your local history
  - Push this commit to the remote repository

You can check that the remote repository was properly updated by refreshing the web page of your repository on GitHub.

- On computer 2
  - Fetch these changes, check you have the proper `script.py` file locally
  - Edit `script.py`:
    ```python
    print("Hello world")
    print("This is me")
    ```
  - Commit these changes
  - Push this commit to the remote repository
- On computer 1
  - Fetch these changes, check that you have the updated `script.py` file locally

Second scenario: collaboration with conflict On the same repository

- On computer 1
  - Edit `script.py`:
    ```python
    print("Hello world")
    print("This is me")
    print("Life should be")
    ```

    - Commit this change
    - Push this commit to the remote repository
- On computer 2
    - Edit `script.py`
      ```
      print("Hello world")
      print("This is me")
      print("Fun for everyone")
      ```
    - Commit these changes
    - Try to push this commit to the remote repository... it should fail!
    - Fetch the changes from the remote repository and resolve conflicts in order to have both lines added
        * See complete instructions on GitHub: "Resolving a merge conflict using the command line"
    - Push this commit to the remote repository
- On computer 1
    - Fetch these changes, check that you have the updated `script.py` file locally

Third scenario: use a Pull Request On the same repository

- On computer 1
    - Create a new branch
    - Edit `script.py`
      ```
      print("Hello world")
      print("This is me")
      print("Life should be")
      print("Fun for everyone")
      print("Hello world")
      print("Come and see")
      ```
    - Commit these changes
    - Push this commit to the remote repository
    - Go to GitHub web page, create a "Pull Request" from this new branch to the main branch
        * See complete instructions on GitHub: "Creating a pull request"
    - Assign your partner as a "Reviewer"
- On computer 2
    - Check out your notifications on GitHub (https://github.com/notifications)
    - Go to the Pull Request
    - Add a comment to the requested changes
        * Go to the "File changed" tab
        * Select multiple lines from the newly added lines
            · Click on liner number "5"
            · Hold "Shift" (or "Maj") and click on line number "6"
        * Move your mouse cursor to code of the last selected line, a blue "+" button should appear on its left
        * Click on the blue button, a new text box should appear
        * Write your comment in the text box about the changes you would like the other to make
        * Once done, click on "Start review"
        * For now, the comment has not been posted. You can make multiple comments before posting them.
        * Once you have made all your comments, click on "Review changes" on the top right, then "Submit review"
- On computer 1
    - Check out your notifications on GitHub
    - Check out the review, reply or apply requested changes
- You can repeat this process as much as you'd like, until both parties are satisfied
- On computer 1
    - Click on "Merge Pull Request"
    - Pull the changes of the main branch
    - Delete the branch of the Pull Request
        * on the remote repository
        * locally

Fourth scenario: rebase vs merge On the same repository

- First read the rebase documentation, especially the first and last sections.
- On computer 1
  - Create a new branch named `sum`
    - create a file `sum.py` with the following content

      ```
      import numpy as np
      import time

      def sum(y):
        res = 0
        for yi in y:
          res += yi
        return res

      def main():
        y = np.random.rand(100000000)
        start = time.time()
        sum(y)
        end = time.time()
        print(f"Elasped time: {end - start}")

      if __name__ == "__main__":
        main()
      ```
  - Check that the code runs
  - Commit these changes
  - Push to the remote repository
- On computer 2
  - Fetch previous changes
  - Create another branch `faster-sum` which starts from the `sum` branch (i.e. checkout `sum` then create the branch)
  - Modify `sum.py` by adding the following function:

    ```
    def faster_sum(y):
      return np.sum(y)
    ```
  - Check that the code still runs
  - Commit these changes
  - Replace the `main` function by

    ```
    def main():
      y = np.random.rand(100000000)

      start = time.time()
      sum(y)
      end = time.time()
      python_time = end - start
      print(f"Python sum: {python_time} s")

      start = time.time()
      faster_sum(y)
      end = time.time()
      numpy_time = end - start
      print(f"Numpy sum: {numpy_time} s")
    ```

```
print(f"Numpy runs {python_time / numpy_time} times faster!")
```

  - Check that the code still runs
  - Commit these changes
  - Push to the remote repository

- On computer 1
  - **On branch sum**
    * Modify the whole file as follows:

      ```
      import numpy as np
      import time

      def sum(y):
        """Return the sum of all elements in array `y`."""
        result = 0
        for yi in y:
          result += yi
        return result

      def main():
        y = np.random.rand(100000000)
        start = time.time()
        sum(y)
        end = time.time()
        print(f"Elasped time: {end - start} seconds")

      if __name__ == "__main__":
        main()
      ```
    * Check that the code runs
    * Commit and push changes

- On computer 2
  - Fetch all recent changes
  - Rebase `faster-sum` on `sum`
    * there should be some conflicts due to modification of the same line by both branches
    * follow git-printed instructions to solve the conflict (`git status` usually helps)
  - Check that the script still runs as expected

- If you have time, you can rerun this scenario by merging `sum` into `faster-sum` instead of rebasing.