UNIVERSITAT DE BARCELONA

Version Control with Git

Analysis and Visualization of Big Data

Franziska Peter and Josep Perelló

Contents - Scientific communication, open science and public participation in research into practice

Sessions V

Version Control with Git. What is it good for? Centralized vs. Distributed VC. Two Basic principles. Hands-on: Getting Started in Git. More nice Features.

Evaluation

Gradual and incremental set of tasks (in class and through Campus Virtual)

Task 1: Data Management Plan Forensics, in group (Tues 9, JPerelló): 10%

Task 2: Sharing code in Github, individual (Wed 10, FPeter): 10%

Task 3: Write an abstract (Mon 15, JPerelló): 10%

Task 4: Create a dashboard (Thu 18, FPeter): 30%

Task 5: Oral presentation, in group (Fri 19, JPerelló + FPeter): 40%

To set a group between 2 and 4. You will work together during the course.

OPEN CODE: Introductory course in Git

Git as a tool for distributed version control

Franziska Peter and Josep Perelló, OpenSystems UB

10 Nov 2021

UNIVERSITAT DE BARCELONA

Before we start: Installation

Recommendation: Use your command line for git.

- Linux/ Mac: first try `git --version`, otherwise install as usual with yum/ apt-get/ zypper/ brew etc.
- Windows: https://git-scm.com/download/win

1. Why version control, why git?

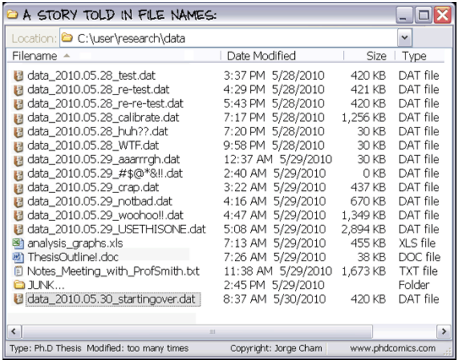
2. Two Basic Principles of Git

3. Getting Started (hands-on)

4. By the way...

Why version control, why git?

Version control systems - Motivation

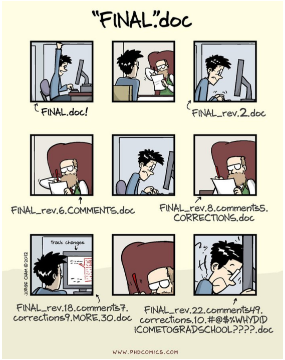


A STORY TOLD IN FILE NAMES:

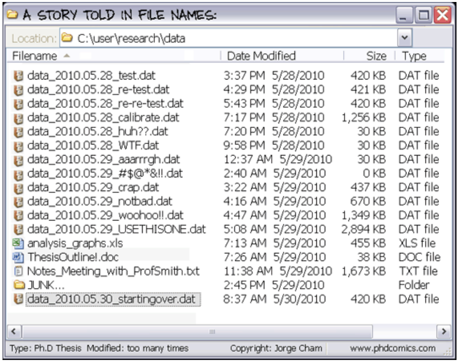
Location: C:\user\research\data

Filename	Date Modified	Size	Type
data_2010.05.28_test.dat	3:37 PM 5/28/2010	420 KB	DAT file
data_2010.05.28_re-test.dat	4:29 PM 5/28/2010	421 KB	DAT file
data_2010.05.28_re-re-test.dat	5:43 PM 5/28/2010	420 KB	DAT file
data_2010.05.28_calibrate.dat	7:17 PM 5/28/2010	1,256 KB	DAT file
data_2010.05.28_huh??.dat	7:20 PM 5/28/2010	30 KB	DAT file
data_2010.05.28_WTF.dat	9:58 PM 5/28/2010	30 KB	DAT file
data_2010.05.29_aaarrrgh.dat	12:37 AM 5/29/2010	30 KB	DAT file
data_2010.05.29_#$@*&!!.dat	2:40 AM 5/29/2010	0 KB	DAT file
data_2010.05.29_crap.dat	3:22 AM 5/29/2010	437 KB	DAT file
data_2010.05.29_notbad.dat	4:16 AM 5/29/2010	670 KB	DAT file
data_2010.05.29_woohoo!!.dat	4:47 AM 5/29/2010	1,349 KB	DAT file
data_2010.05.29_USETHISONE.dat	5:08 AM 5/29/2010	2,894 KB	DAT file
analysis_graphs.xls	7:13 AM 5/29/2010	455 KB	XLS file
ThesisOutline!.doc	7:26 AM 5/29/2010	38 KB	DOC file
Notes_Meeting_with_ProfSmith.txt	11:38 AM 5/29/2010	1,673 KB	TXT file
JUNK...	2:45 PM 5/29/2010		Folder
data_2010.05.30_startingover.dat	8:37 AM 5/30/2010	420 KB	DAT file

Type: Ph.D Thesis Modified: too many times Copyright: Jorge Cham www.phdcomics.com


"FINAL".doc
FINAL.doc!
FINAL_rev.2.doc
FINAL_rev.6.COMMENTS.doc
FINAL_rev.8.comments5.CORRECTIONS.doc
Track changes
JORGE CHAM © 2012
FINAL_rev.18.comments7.corrections9.MORE.30.doc
FINAL_rev.22.comments49.corrections.10.#@$%WHYDIDICOMETOGRADSCHOOL????.doc
WWW.PHDCOMICS.COM

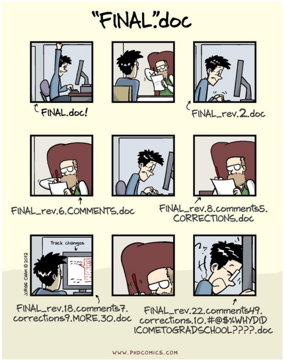
Version control systems - Motivation


A STORY TOLD IN FILE NAMES:

Location: C:\user\research\data

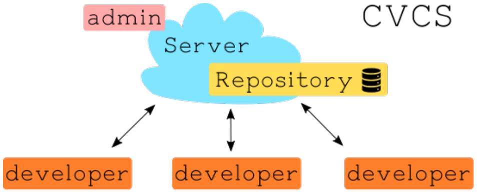
Filename	Date Modified	Size	Type
data_2010.05.28_test.dat	3:37 PM 5/28/2010	420 KB	DAT file
data_2010.05.28_re-test.dat	4:29 PM 5/28/2010	421 KB	DAT file
data_2010.05.28_re-re-test.dat	5:43 PM 5/28/2010	420 KB	DAT file
data_2010.05.28_calibrate.dat	7:17 PM 5/28/2010	1,256 KB	DAT file
data_2010.05.28_huh??.dat	7:20 PM 5/28/2010	30 KB	DAT file
data_2010.05.28_WTF.dat	9:58 PM 5/28/2010	30 KB	DAT file
data_2010.05.29_aaarrrgh.dat	12:37 AM 5/29/2010	30 KB	DAT file
data_2010.05.29_#$@*&!!.dat	2:40 AM 5/29/2010	0 KB	DAT file
data_2010.05.29_crap.dat	3:22 AM 5/29/2010	437 KB	DAT file
data_2010.05.29_notbad.dat	4:16 AM 5/29/2010	670 KB	DAT file
data_2010.05.29_woohoo!!.dat	4:47 AM 5/29/2010	1,349 KB	DAT file
data_2010.05.29_USETHISONE.dat	5:08 AM 5/29/2010	2,894 KB	DAT file
analysis_graphs.xls	7:13 AM 5/29/2010	455 KB	XLS file
ThesisOutline!.doc	7:26 AM 5/29/2010	38 KB	DOC file
Notes_Meeting_with_ProfSmith.txt	11:38 AM 5/29/2010	1,673 KB	TXT file
JUNK...	2:45 PM 5/29/2010		Folder
data_2010.05.30_startingover.dat	8:37 AM 5/30/2010	420 KB	DAT file

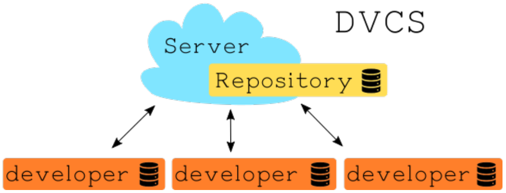
Type: Ph.D Thesis Modified: too many times Copyright: Jorge Cham www.phdcomics.com


"FINAL".doc
FINAL.doc!
FINAL_rev.2.doc
FINAL_rev.6.COMMENTS.doc
FINAL_rev.8.comments5.CORRECTIONS.doc
Track changes
FINAL_rev.18.comments7.corrections9.MORE.30.doc
FINAL_rev.22.comments49.corrections.10.#@$%WHYDID ICOMETOGRADSCHOOL????.doc
JORGE CHAM © 2012
WWW.PHDCOMICS.COM

a good VCS should:
have a log book / allow work in parallel with others / allow work in parallel on different versions / (external) backup / be simple / be speedy & efficient

Centralized vs. Distributed Version control systems

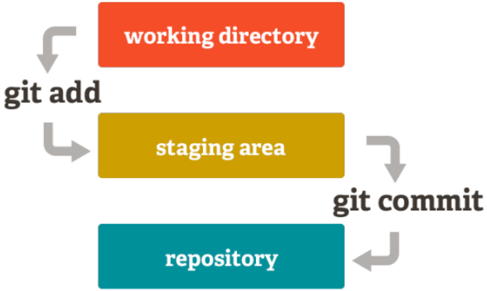




Two Basic Principles of Git

Everyday Mantra: stage and commit (local)

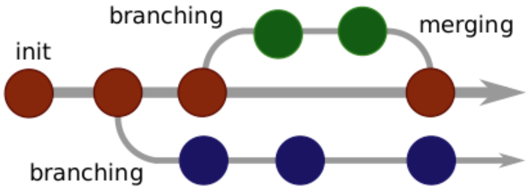
The three states of git: **modified - staged - committed**



```
git add .
git commit -m "implemented calculation of mu"
git push origin master
git pull
```

Working in parallel on different versions: branching

Nonlinear development:
working on several versions **in parallel**, with the possibility of **reuniting** the different resulting versions



```
git checkout -b feature
git checkout master
git merge feature
```

Getting Started (hands-on)

Create your first own LOCAL git repository

Create a folder for today's course. Create a subfolder called `hello_octocat`. Run the following commands in your favourite shell via command line from within folder `hello_octocat`:

- `git init` now your subfolder is a repository

info Linux/ Mac: `ls -a`   Windows: `dir a`

info `git status` which branch are you on?, staged/committed?

info `git log` show all commits

- create file with content inside `hello_octocat/` (e.g. `hello_world.py`), e.g. with vim or nano

Create your first own LOCAL git repository

Create a folder for today's course. Create a subfolder called `hello_octocat`. Run the following commands in your favourite shell via command line from within folder `hello_octocat`:

- `git init` now your subfolder is a repository

info Linux/ Mac: `ls -a`   Windows: `dir a`

info `git status` which branch are you on?, staged/committed?

info `git log` show all commits

- create file with content inside `hello_octocat/` (e.g. `hello_world.py`), e.g. with vim or nano
- `git add .`
- `git commit -m "added file hello_world.py"`

Create your first own LOCAL git repository

Create a folder for today's course. Create a subfolder called `hello_octocat`. Run the following commands in your favourite shell via command line from within folder `hello_octocat`:

- ○ `git init` now your subfolder is a repository

info Linux/ Mac: `ls -a`   Windows: `dir a`

info `git status` which branch are you on?, staged/committed?

info `git log` show all commits

- ○ create file with content inside `hello_octocat/` (e.g. `hello_world.py`), e.g. with vim or nano
- ○ `git add .`
- ○ `git commit -m "added file hello_world.py"`
- ○ change file, e.g. `hello_world.py`, then run `git add .` , `git diff HEAD` , and `git commit -m "removed typo"`

Publish (?) your repository on GitHub

- Sign Up on https://github.com/
    - create a **token** on github.com under Settings>Developer settings>Personal access tokens

Publish (?) your repository on GitHub

- Sign Up on https://github.com/
  - create a **token** on github.com under Settings>Developer settings>Personal access tokens
- "upload" your local repository to github:
  - Either use `git remote add origin https://github.com/Chaotique/learn-git-a-bit.git` with the token as password
  - or use git remote add origin https://**<token>**@github.com/Chaotique/learn-git-a-bit.git
  - `git push -u origin master`

Publish (?) your repository on GitHub

- Sign Up on https://github.com/
  - create a **token** on github.com under Settings>Developer settings>Personal access tokens
- "upload" your local repository to github:
  - Either use `git remote add origin https://github.com/Chaotique/learn-git-a-bit.git` with the token as password
  - or use git remote add origin https://**<token>**@github.com/Chaotique/learn-git-a-bit.git
  - `git push -u origin master`
- alternative: create a repo on github and clone from there:
  - `git clone https://github.com/Chaotique/test-repo.git`
    And then git remote **set-url** origin https://**<token>**@github.com/Chaotique/test-repo.git

Working with remotes

info `git remote -v` list remotes with url

- `git remote add foo <url>` add a remote with alias "foo"
- `git remote set-url foo <url>` change url of remote foo
- `git push origin master` send updates to origin (origin is the default alias for the default remote)
- `git clone <url>` download complete copy of a repo (with full history) from a platform and make it a remote
- `git pull` = `git fetch` + `git merge` get updates

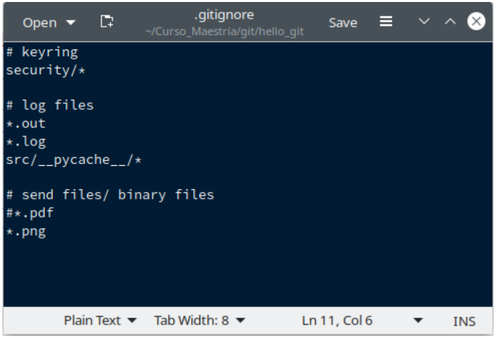
Working in parallel: branching

- `git branch developer` create a branch called "developer"
- `git checkout developer` switch to branch "developer"
- `git checkout -b developer` create a branch called "developer" and switch to it
- `git checkout master` and `git merge developer` switch to master and merge developer branch into master branch
- if necessary, resolve merge conflicts
- `git branch -d developer` delete branch "developer" after successful merge

By the way…

By the way... the first thing to do after `git init`

Adding a `.gitignore` file (using glob patterns) to your `hello_octocat/` folder (or any of its subfolders) makes git ignore the specified files and folders.



```
# keyring
security/*

# log files
*.out
*.log
src/__pycache__/*

# send files/ binary files
#*.pdf
*.png
```

By the way... know the difference(s)

You can compare any version on any branch with any version in any state. Most typical examples:

- `git diff`: working directory vs. staged
- `git diff --staged`: staged changes vs. last commit (HEAD)
- `git diff HEAD`: HEAD vs. working directory

For any other versions: you'll find out when you need it :P
Also: Try `git log -p -3` to see log & changes from the last three commits.

By the way... how to undo things? DANGER

DANGER

Many steps can be undone on git. But be careful not to delete anything important!

- `git restore --staged example.py` to undo staging (> Git version 2.23.0), `git restore example.py` throws away your modifications, so careful!
- `git commit --amend` to add sth to the commit (use only locally!!)
- `git checkout -- example.py` to throw away all modifications you did since the last commit - DANGER!

By the way... how to delete or move stuff?

You can't simply remove a file, git will notice (tracked files), use `git rm example.py` instead.

Same story about moving files. Either use

```
mv example.py src
git add src/example.py
git rm example.py
```

or instead simply do

```
git mv example.py src
```

What’s more?

- **tagging**: give special commits a special name
- **issues**: mark bugs and later refer to them in the fixing commit
- **stashing**: hold modifications (staged or unstaged, but not yet committed) in the air to make other operations first, then pop them back in
- **forks and pull requests**: branch from other peoples projects/repositories and later ask the to accept your proposed changes
- **rebase**: do clean up work on your forest of commits, or run messing up everything completely (experts command)

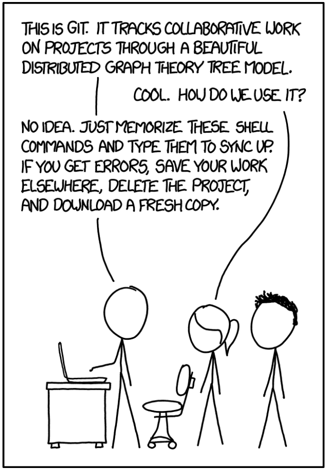
Further reading, Tutorials, Cheat Sheets

📕 Scott Chacon, Ben Straub (2021)
Pro Git
https://git-scm.com/book/en/v2

Tutorials:
https://try.github.io
https://www.atlassian.com/git/tutorials

Cheat sheets: (several languages)
https://www.git-tower.com/blog/git-cheat-sheet/
https://training.github.com/

Have a lot of fun and joy with git!


THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.

And don't forget:
If you upload the codes that you wr(i/o)te during the two weeks and you **fork** them with me (Chaotique), you get some extra credit for your final degree for this assignment!

Earn some credit!

Task 2: Sharing code in Github, individual (Wed 10, FPeter): 10%

1. Upload some code (git add. , git commit, git push)

2. Make changes to this code and upload them (same way)

3. Invite me as a collaborator to your repository

Earn some credit!

Task 2: Sharing code in Github, individual (Wed 10, FPeter): 10%

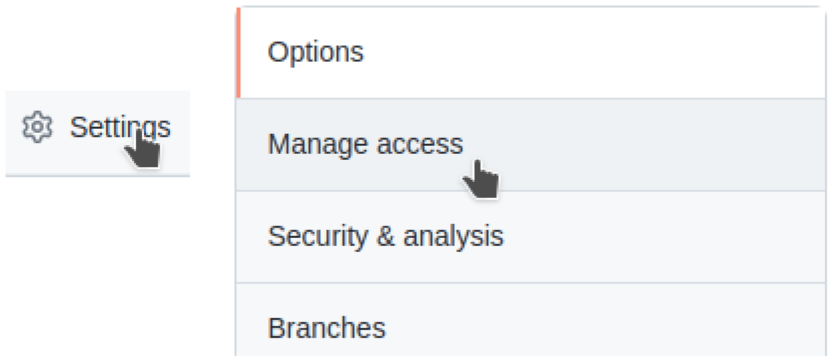
1. Upload some code (git add. , git commit, git push)
2. Make changes to this code and upload them (same way)
3. Invite me as a collaborator to your repository

In github, on your repositories page, go to settings.
Then follow next screenshots...

Earn some credit!

Task 2: Sharing code in Github, individual (Wed 10, FPeter): 10%

1. Upload some code (git add. , git commit, git push)
2. Make changes to this code and upload them (same way)
3. Invite me as a collaborator to your repository

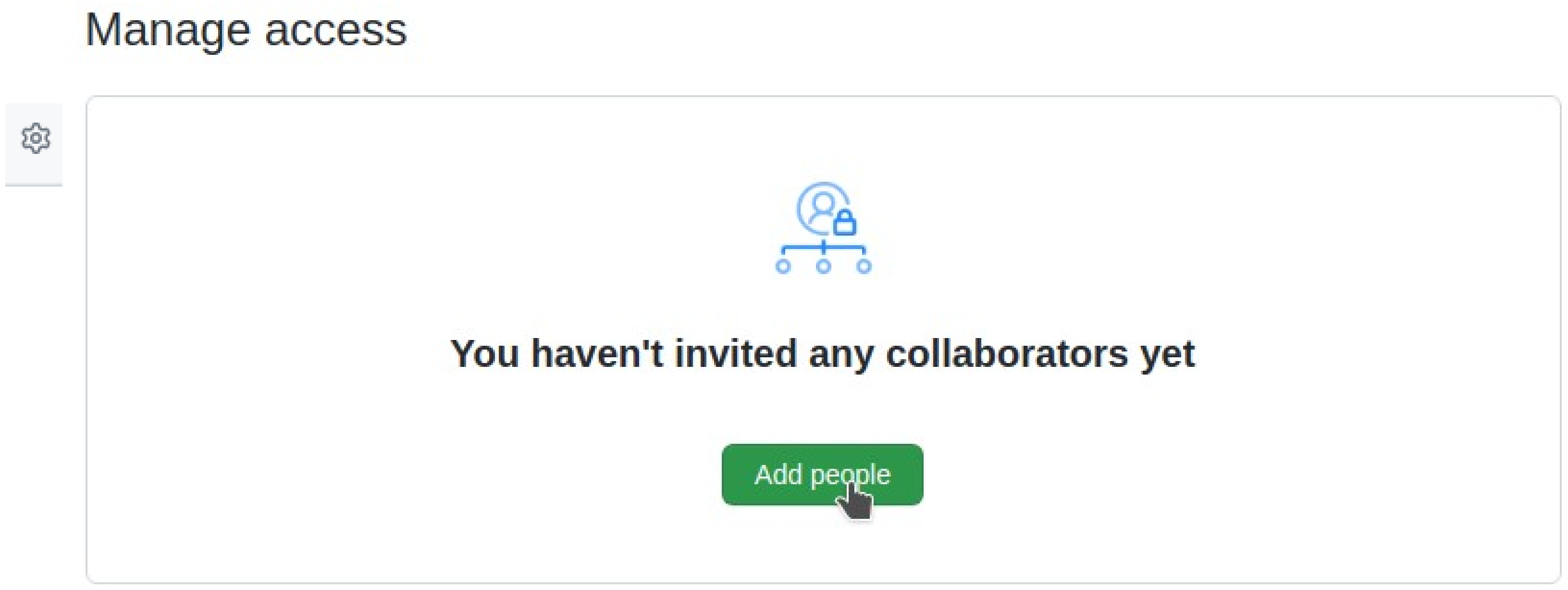


6

Earn some credit!

Task 2: Sharing code in Github, individual (Wed 10, FPeter): 10%

1. Upload some code (git add. , git commit, git push)
2. Make changes to this code and upload them (same way)
3. Invite me as a collaborator to your repository

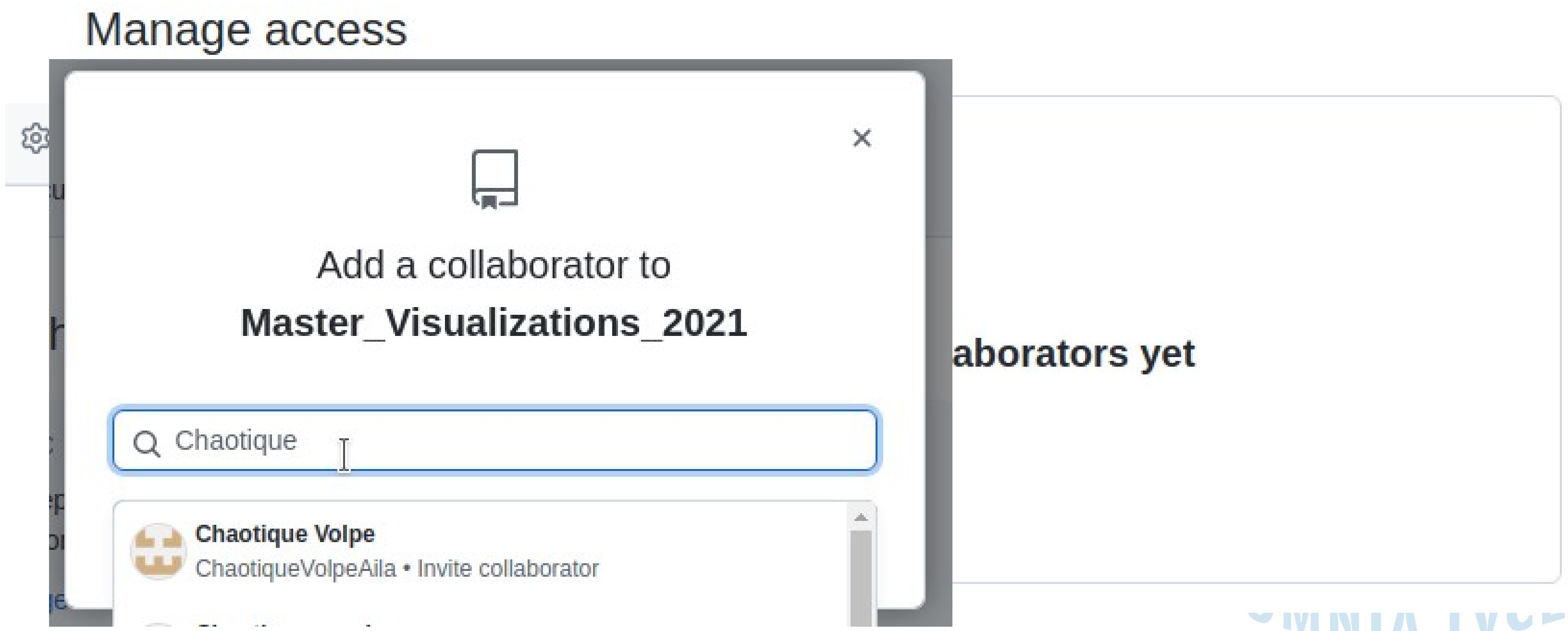

Manage access

You haven't invited any collaborators yet

Earn some credit!

Task 2: Sharing code in Github, individual (Wed 10, FPeter): 10%

1. Upload some code (git add. , git commit, git push)
2. Make changes to this code and upload them (same way)
3. Invite me as a collaborator to your repository


Manage access

Add a collaborator to
Master_Visualizations_2021

Chaotique

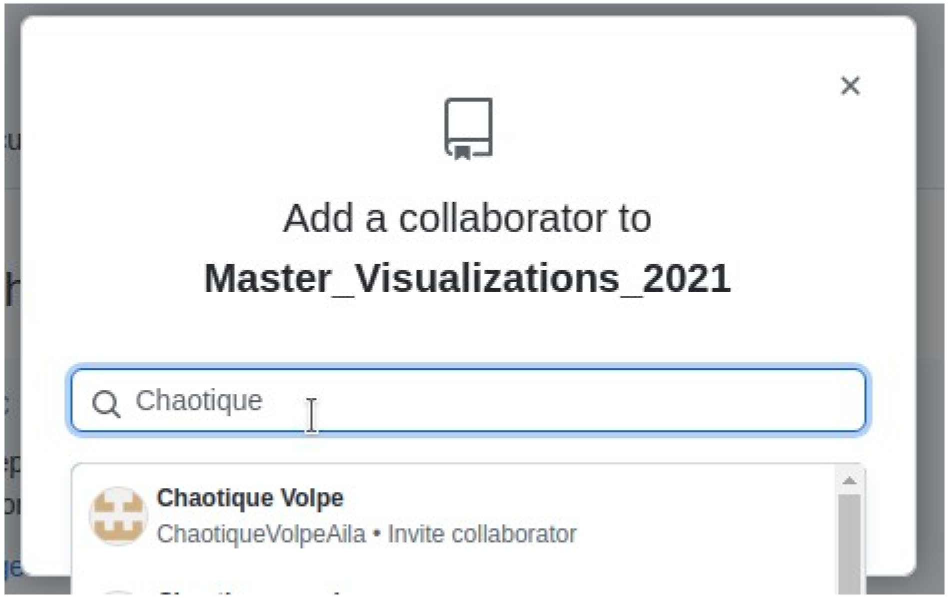
Chaotique Volpe
ChaotiqueVolpeAila • Invite collaborator

aborators yet

Earn some credit!

Task 2: Sharing code in Github, individual (Wed 10, FPeter): 10%

1. Upload some code (git add. , git commit, git push)
2. Make changes to this code and upload them (same way)
3. Invite me as a collaborator to your repository


Add a collaborator to
Master_Visualizations_2021

Chaotique

Chaotique Volpe
ChaotiqueVolpeAila • Invite collaborator

That's me,
Chaotique