Le versioning, c'est quoi ?

Pourquoi, comment, et sur quoi?

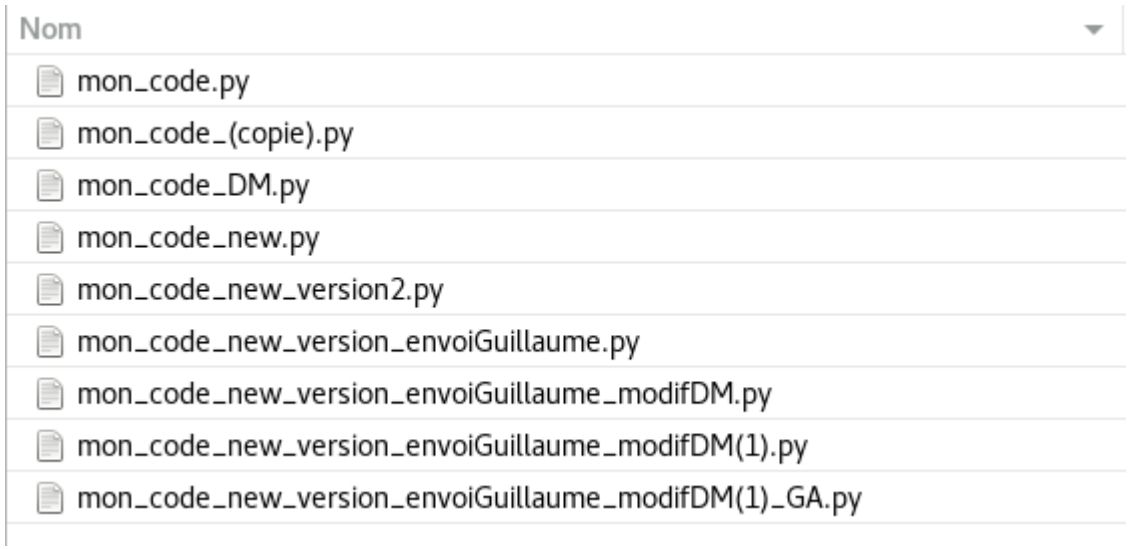
DataClub CRPN 08/10/2021

David Meunier
Centre IRM – INT @ CERIMED
david.meunier@univ-amu.fr

Why versionning ?

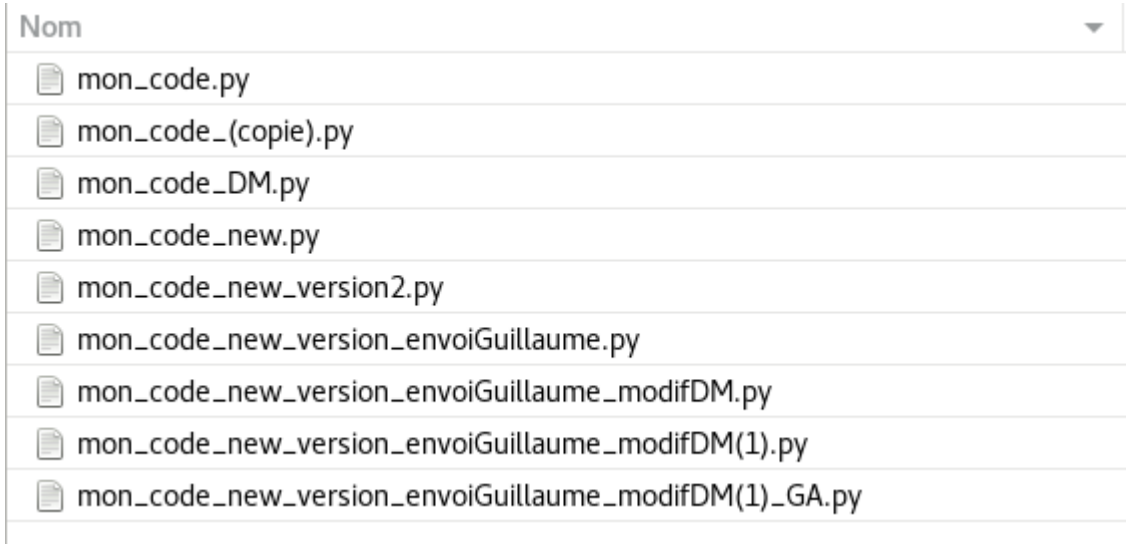
Context

When you code with someone else (your PhD director, a collaborator, a student) :



Context

When you code with someone else (your PhD director, a collaborator, a student) :



Solution : versionning

Storage of the successive versions of a project can be retrieved from a single place.

What to version ?

What to version ?

- For anything readable (and relatively small) : **git**

  Git is not restricted to source code : also website, etc tutorial

- For anything bigger : **git-annex** and more adequaly **datalad**

How to version ?

How to version source code: Git

What is git ?

→ successor of previous versioning system : CVS and SVN

→ git allows for multiple people to work together on a project : collaborative development

→ git also allows to make branch (i.e. each developer works on a subpart of the project) and then allows ‘automatic’ merging of all the branches

How to use git ?

→ Most common instanciation: github (cf follow-up) and more interestingly gitlab

→ gitlab itself is a software that can be installed on a server (for example https://gitlab.com/icm-institute)

How to version data: Git-annex and datalad

And versionning other types of files?

For data, there exists a solution for « bigger » files , typically binary data called **git-annex**

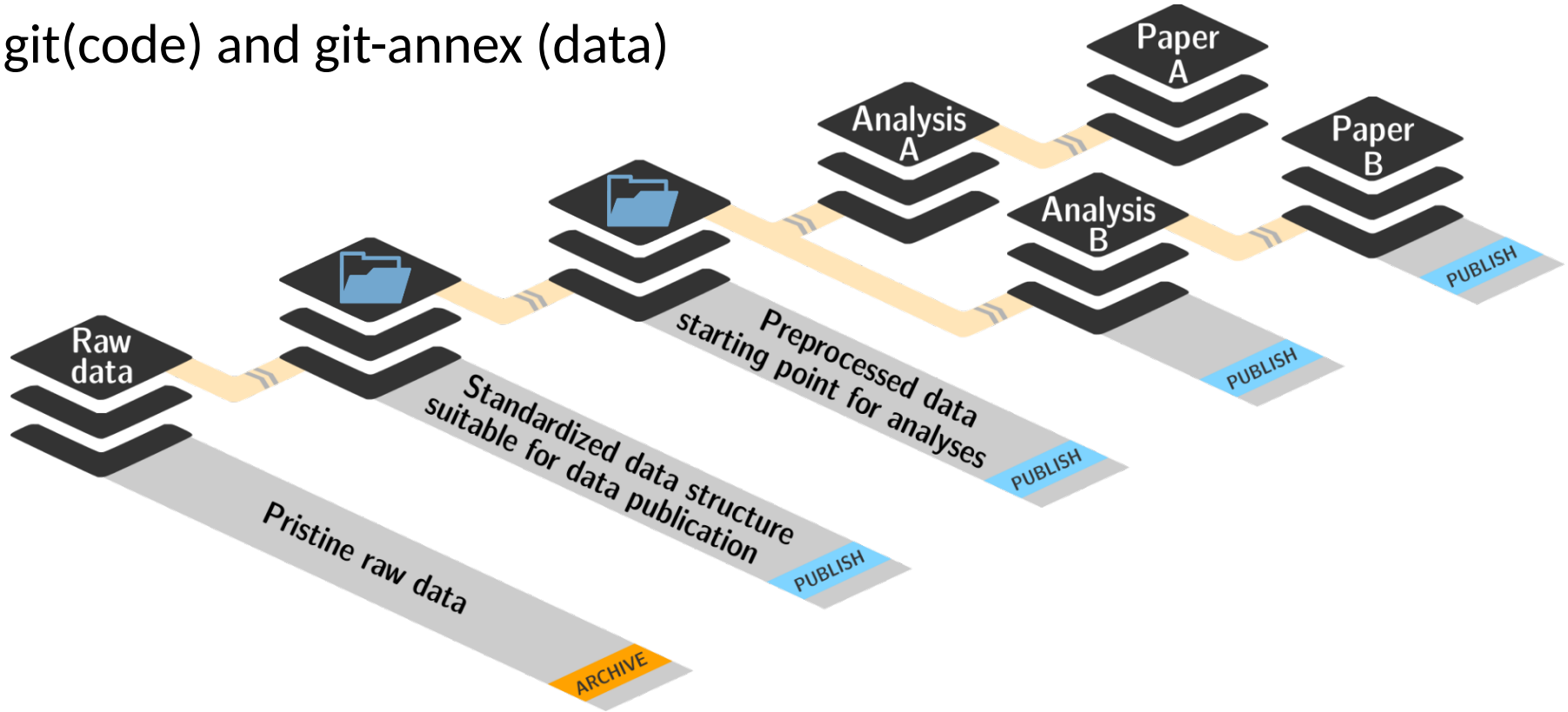
How to version data: Git-annex and datalad

And versionning other types of files?

For data, there exists a solution for « bigger » files , typically binary data called **git-annex**

Datalad :

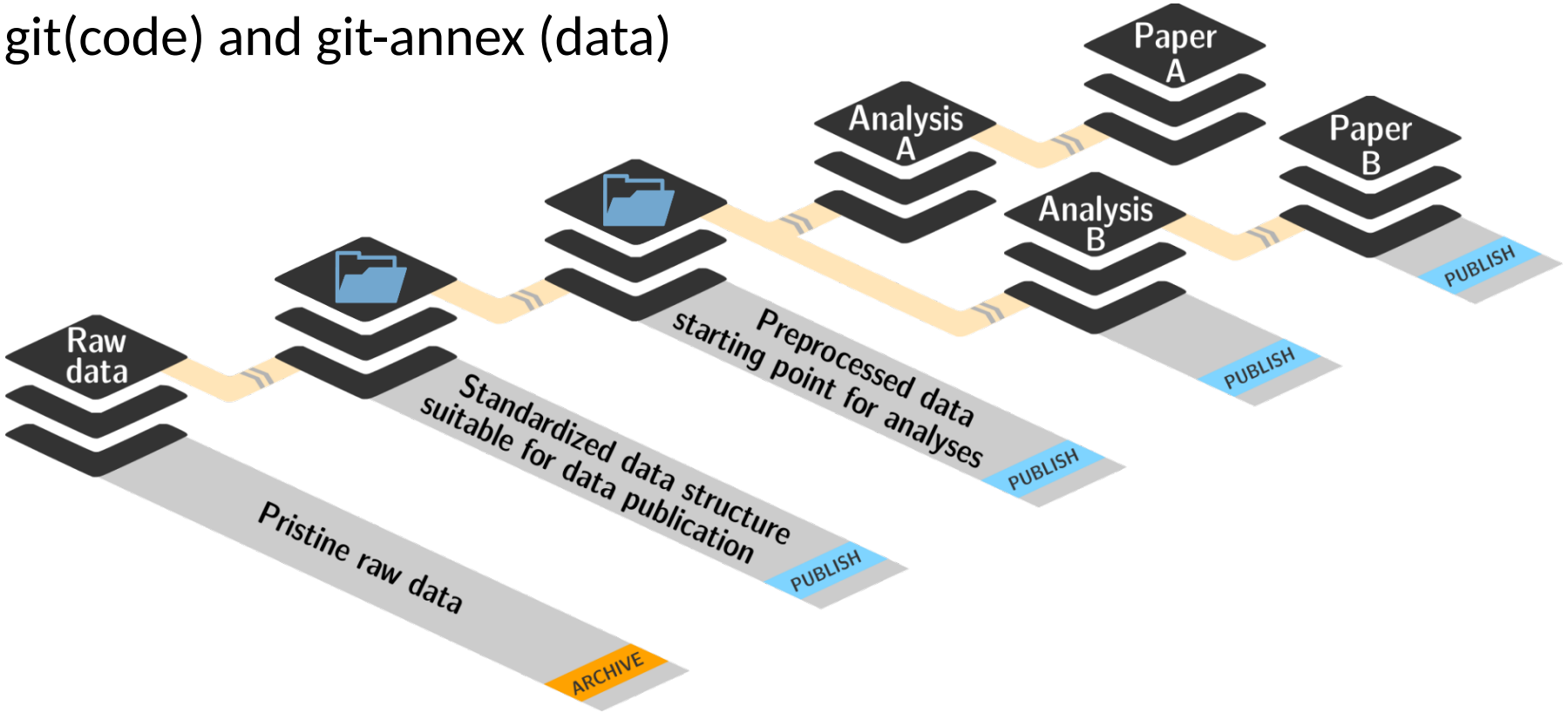
build on top of git(code) and git-annex (data)



How to version data: Git-annex and datalad

Datalad :

build on top of git(code) and git-annex (data)



Some other notion related to data versionning :

- Provenance tracking : store all information (software/container version, command line, OS characteristics) between dataset version

- FAIR Compliance : assign DOI (e.g. Zenodo, or OpenNeuro / PublicNeuro) to publish dataset

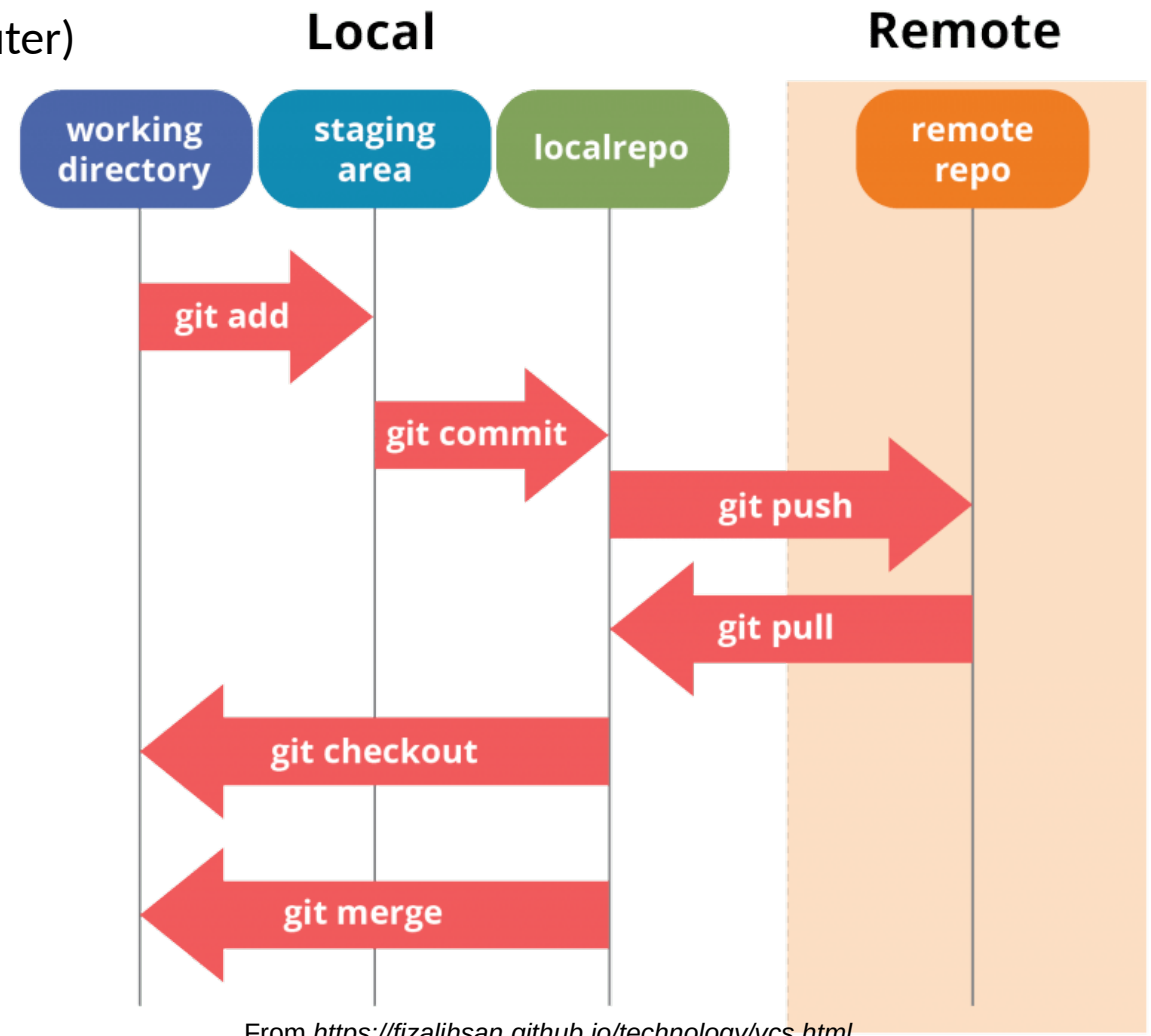
How to version ? Git principles

Git

Basic principles:

→ a repository (a repo) is a place on a **remote** (i.e. not on your computer) where the « project » is stored.

→ The project on the server is the « good » version, and the « **local** » copy to be modified and ongoing work



From *https://fizalihsan.github.io/technology/vcs.html*

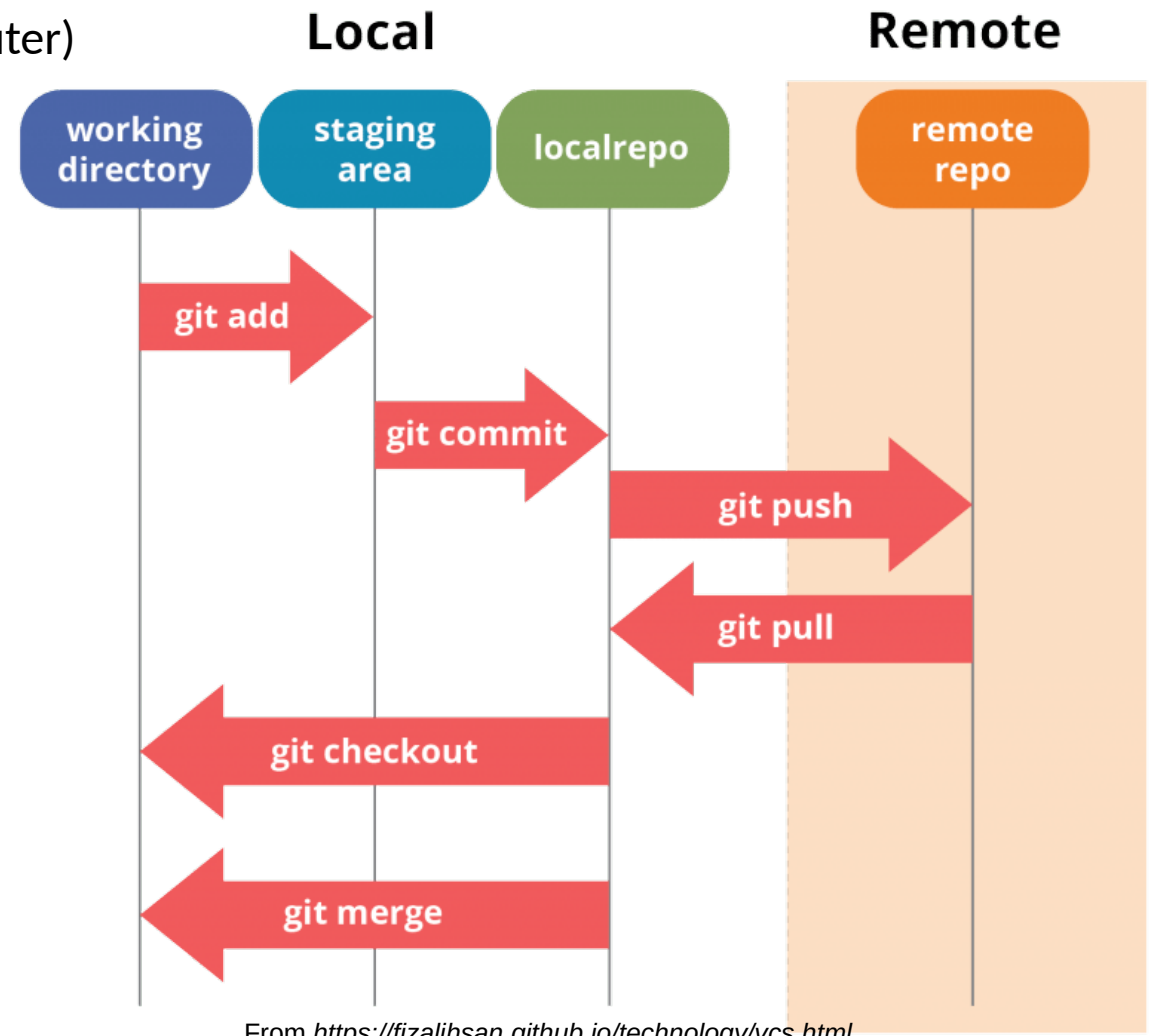
Git

Basic principles:

→ a repository (a repo) is a place on a **remote** (i.e. not on your computer) where the « project » is stored.

→ The project on the server is the « good » version, and the « **local** » copy to be modified and ongoing work

By default, the remote is called « origin »



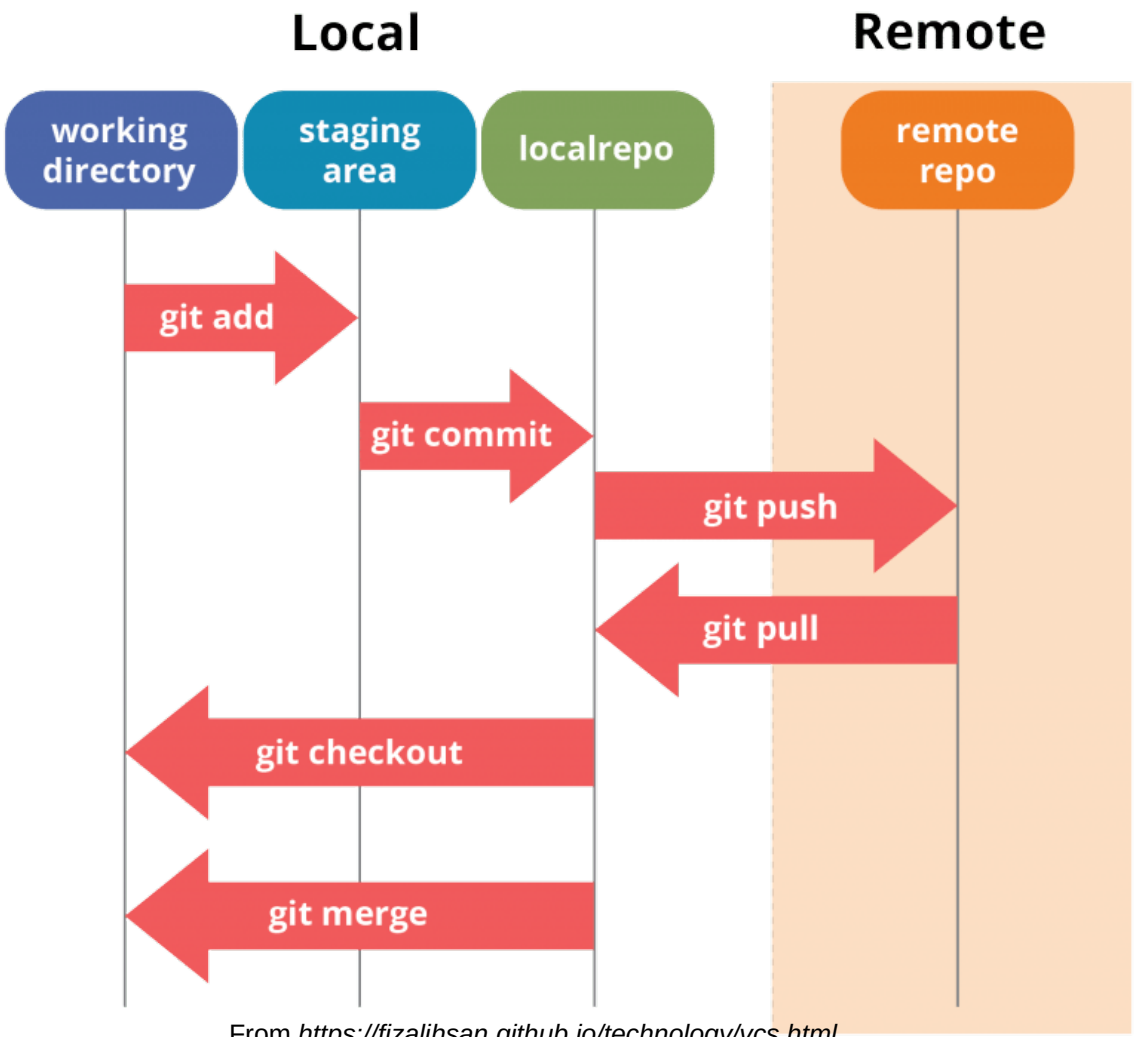
From *https://fizalihsan.github.io/technology/vcs.html*

Git

Basic operations:

`$ git clone repo_adress/my_project.git`

You have now a local version of *my_project*



From *https://fizalihsan.github.io/technology/vcs.html*

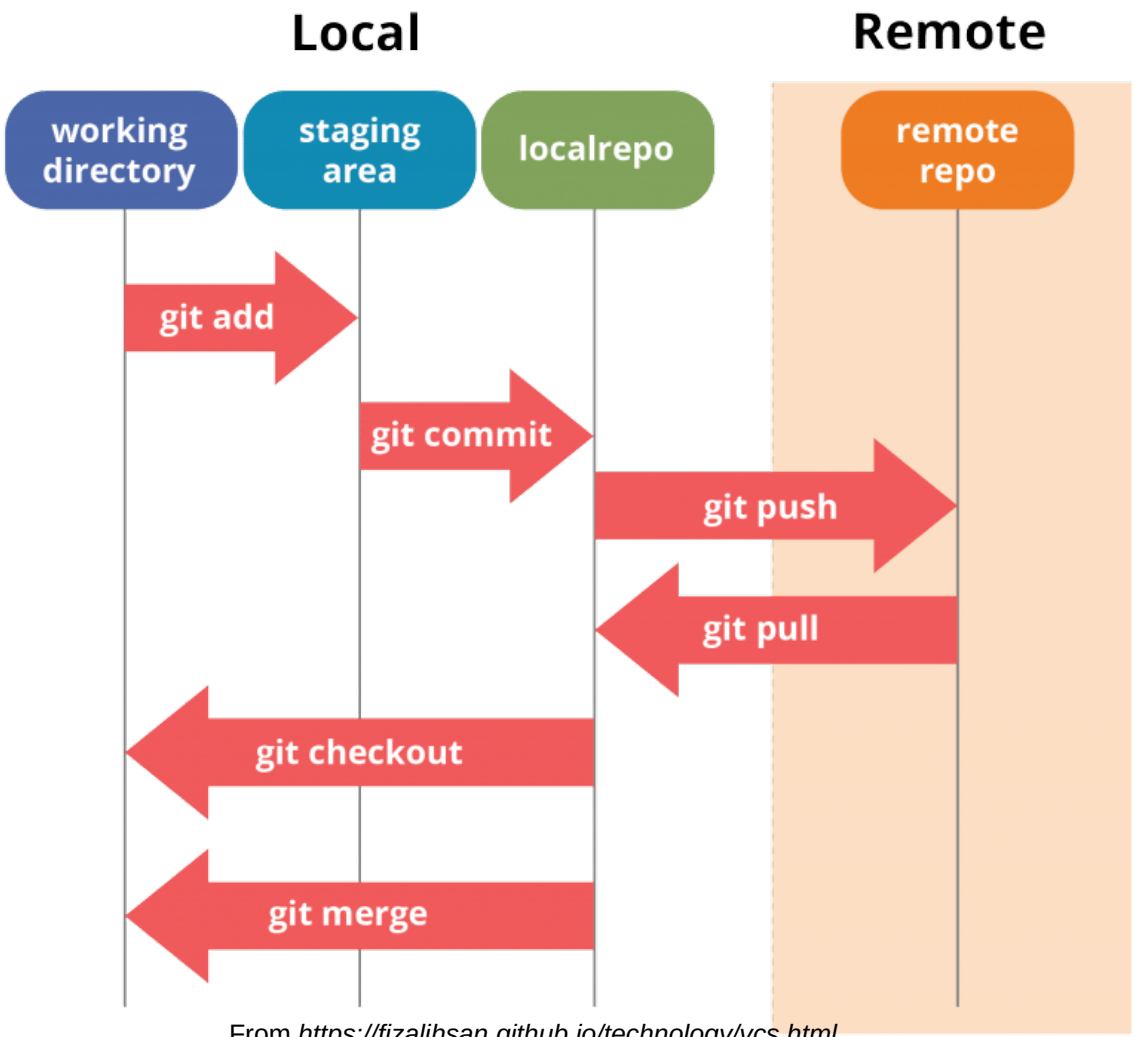
Git

Basic operations:

```
$ git clone repo_adress/my_project.git
```

You have now a local version of *my_project*

(side note : be careful, on github now, *https* will require 2 Factor Authentification 2-FA, *ssh* with public/private key)



From *https://fizalihsan.github.io/technology/vcs.html*

Git

Basic operations:

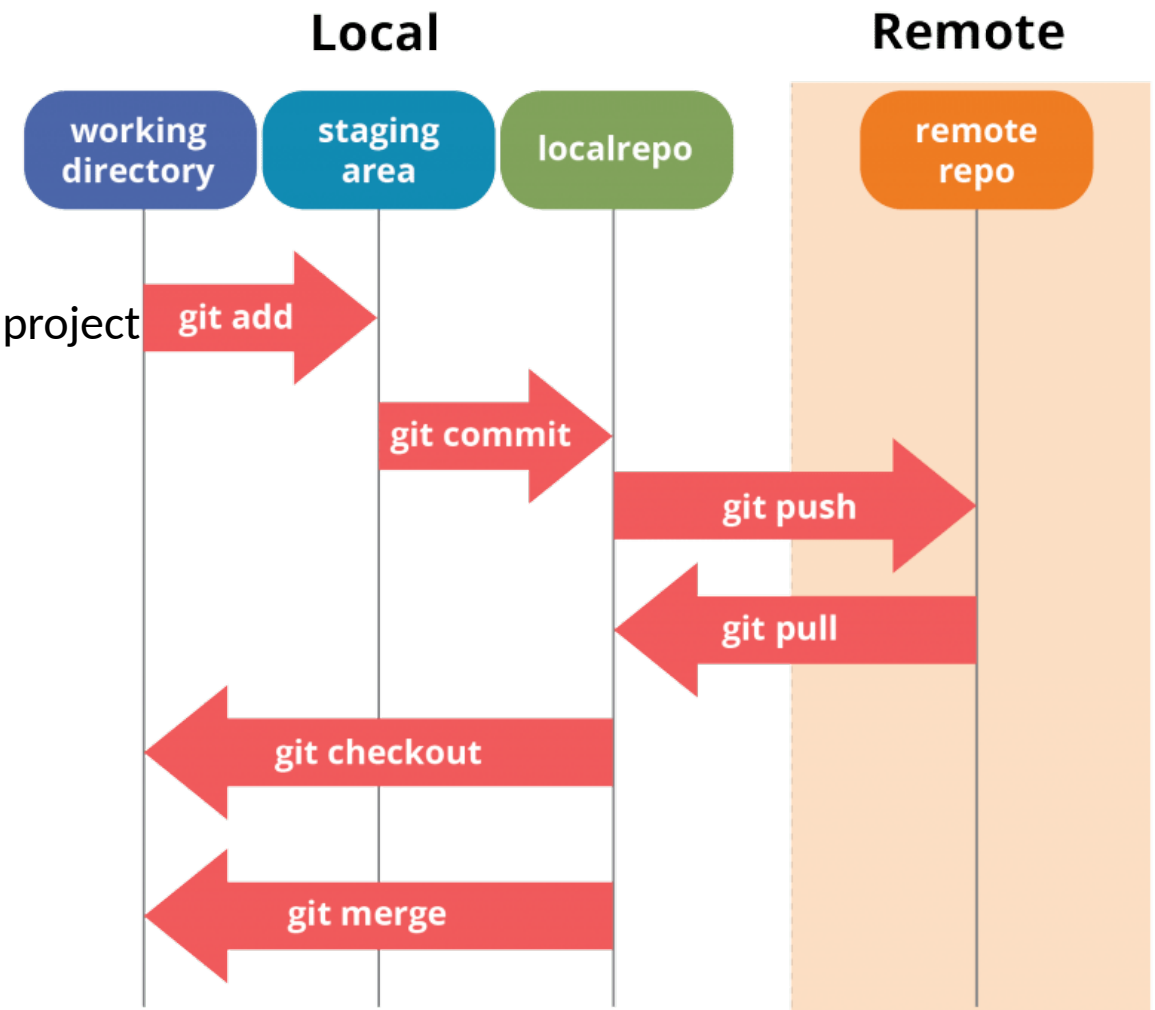
`$ git clone` *`repo_adress/my_project.git`*

You have now a local version of *my_project*

→ do some modifications on the project ;

→ once you are happy with the modifications you made locally :

`$ git add new_file.py` (only when a new file is added to the project not needed if an existing file is modified)



From *https://fizalihsan.github.io/technology/vcs.html*

Git

Basic operations:

`$ git clone repo_adress/my_project.git`

You have now a local version of *my_project*

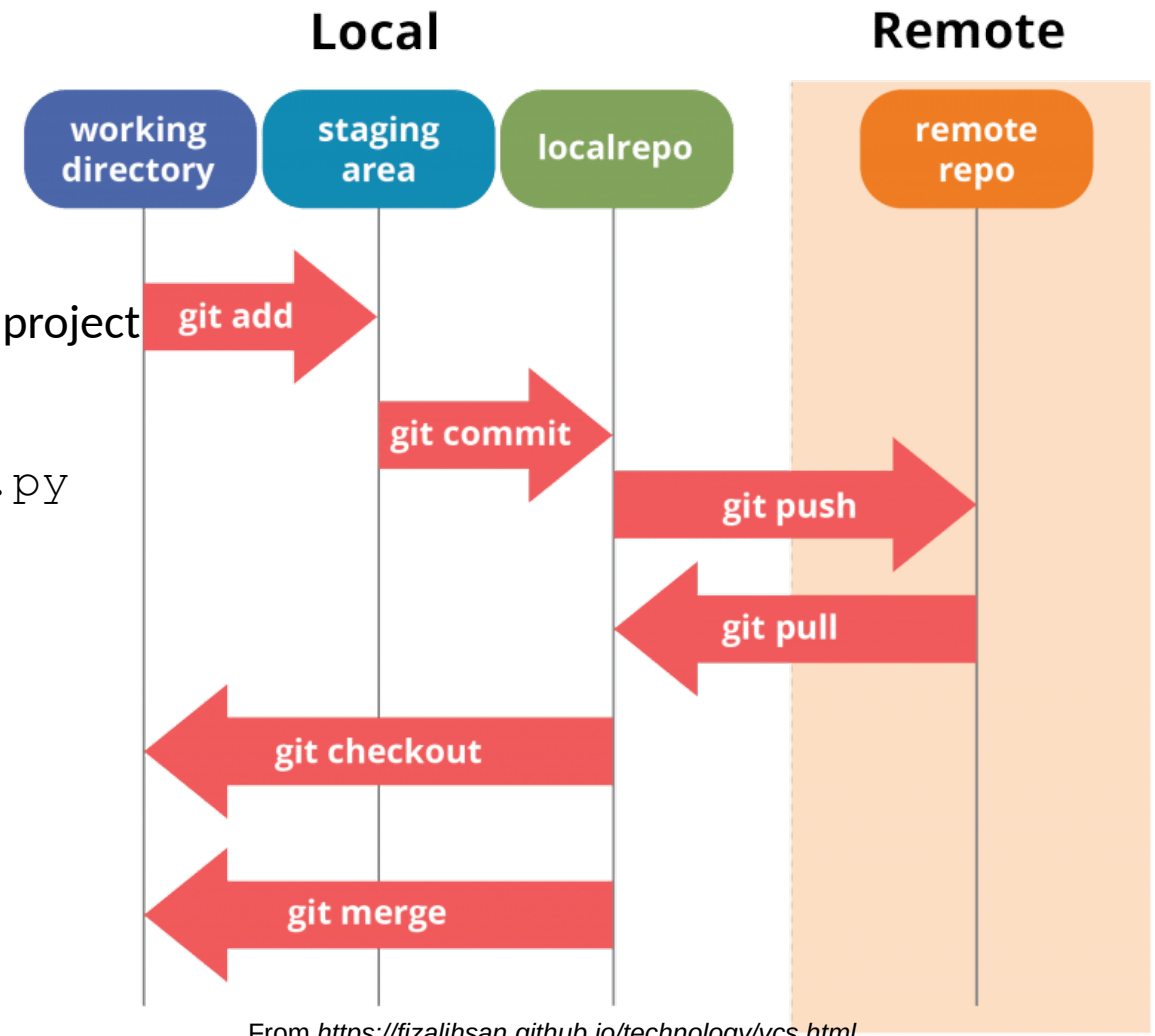
→ do some modifications on the project ;

→ once you are happy with the modifications you made locally :

`$ git add new_file.py` (only when a new file is added to the project not needed if an existing file is modified)

`$ git commit –m 'my new modifications' my_file.py`

(describe what you did to the project)



From *https://fizalihsan.github.io/technology/vcs.html*

Git

Basic operations:

```
$ git clone repo_adress/my_project.git
```

You have now a local version of *my_project*

→ do some modifications on the project ;

→ once you are happy with the modifications you made locally :

`$ git add new_file.py` (only when a new file is added to the project not needed if an existing file is modified)

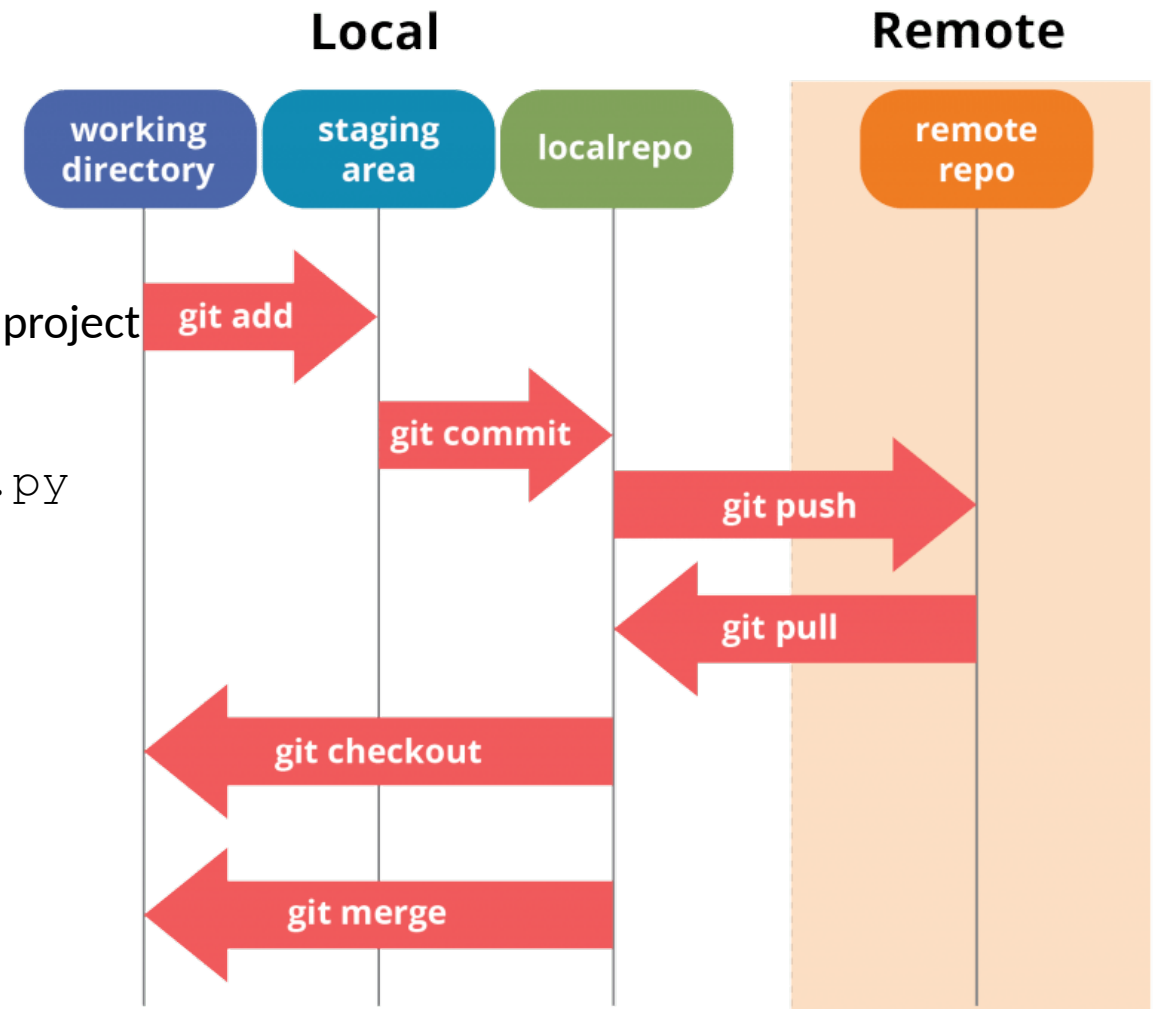
```
$ git commit –m 'my new modifications' my_file.py
```

(describe what you did to the project)

```
$ git push origin master
```

(push your version to become the new standard on the repo)

NB : default branch is named *master*



From *https://fizalihsan.github.io/technology/vcs.html*

Git

Basic operations:

```
$ git clone repo_adress/my_project.git
```

You have now a local version of *my_project*

→ do some modifications on the project ;

→ once you are happy with the modifications you made locally :

`$ git add new_file.py` (only when a new file is added to the project not needed if an existing file is modified)

```
$ git commit –m 'my new modifications' my_file.py
```

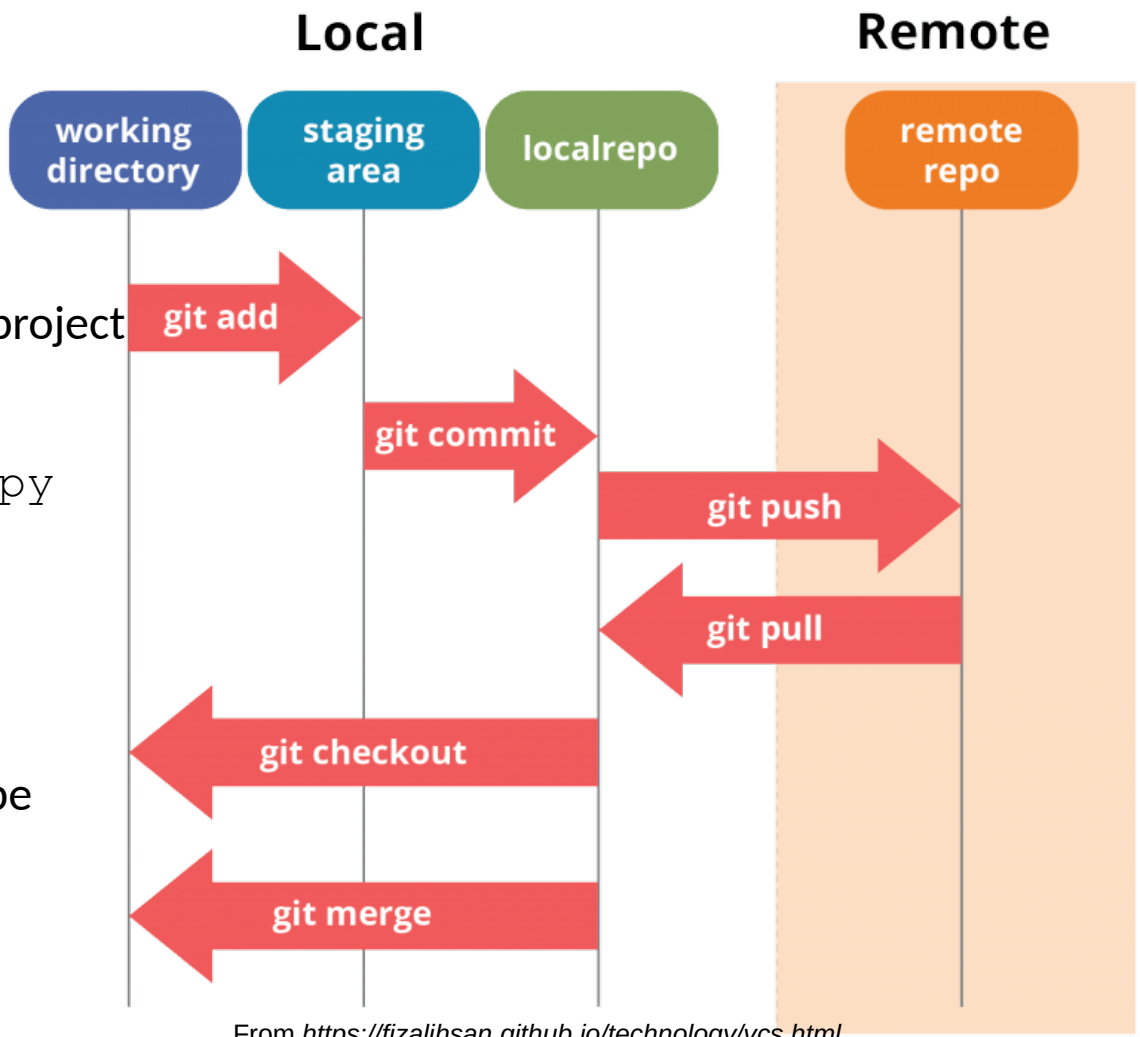
(describe what you did to the project)

```
$ git push origin master
```

(push your version to become the new standard on the repo)

Before pushing your version, the newest version on the repo should be retrieved by

$ git pull origin master



From *https://fizalihsan.github.io/technology/vcs.html*

Git

I made a mistake:

$ git status

Will show the « status » of the files in a local git directory

```
(base) 2 [15:24:46] Je n'ai pas de nom !@CRI-103-DE01: ~/miniconda3/Packages/neuropycon_demo/OpenfMRI_ds000117  $ git status
Sur la branche add_pybids
Fichiers non suivis:
  (utilisez "git add <fichier>..." pour inclure dans ce qui sera validé)

        02-plot_preprocessing_pybids.py
        10_events_inverse_stc_old.py
        10_events_inverse_stc_old2.py
        __pycache__/
        params_testDM.json
        ../OpenfMRI_ds000117_pybids/log.txt
        ../OpenfMRI_ds000117_pybids/params_nocli.json
        ../OpenfMRI_ds000117_pybids/prep_trans.py
        ../OpenfMRI_ds000117_pybids/trns.txt

aucune modification ajoutée à la validation mais des fichiers non suivis sont présents (utilisez "git add" pour les suivre)
```

Git

I made a mistake:

I make some modifications on a « followed » file (i.e. a file that have been added to the git project)

```
(base) 3 [15:24:52] Je n'ai pas de nom !@CRI-103-DE01: ~/miniconda3/Packages/neuropycon_demo/OpenfMRI_ds000117  $ git status
Sur la branche add_pybids
Modifications qui ne seront pas validées :
  (utilisez "git add <fichier>..." pour mettre à jour ce qui sera validé)
  (utilisez "git checkout -- <fichier>..." pour annuler les modifications dans la copie de travail)

        modifié :         ../OpenfMRI_ds000117_pybids/02-03-plot_preprocessing_inverse_pybids.py

Fichiers non suivis:
  (utilisez "git add <fichier>..." pour inclure dans ce qui sera validé)

        02-plot_preprocessing_pybids.py
        10_events_inverse_stc_old.py
        10_events_inverse_stc_old2.py
        __pycache__/
        params_testDM.json
        ../OpenfMRI_ds000117_pybids/log.txt
        ../OpenfMRI_ds000117_pybids/params_nocli.json
        ../OpenfMRI_ds000117_pybids/prep_trans.py
        ../OpenfMRI_ds000117_pybids/trns.txt

aucune modification n'a été ajoutée à la validation (utilisez "git add" ou "git commit -a")
```

Git

I made a mistake:

`$ git stash`

You erase all the modications that have NOT been commited so far (i.e. you go back to the latest pulled version)

```
(base) 4 [15:26:04] Je n'ai pas de nom !@CRI-103-DE01: ~/miniconda3/Packages/neuropycon_demo/OpenfMRI_ds000117  $ git stash
Copie de travail et état de l'index sauvegardés dans WIP on add_pybids: a7c6046 bug
HEAD est maintenant à a7c6046 bug
(base) 5 [15:27:04] Je n'ai pas de nom !@CRI-103-DE01: ~/miniconda3/Packages/neuropycon_demo/OpenfMRI_ds000117  $ git status
Sur la branche add_pybids
Fichiers non suivis:
  (utilisez "git add <fichier>..." pour inclure dans ce qui sera validé)

        02-plot_preprocessing_pybids.py
        10_events_inverse_stc_old.py
        10_events_inverse_stc_old2.py
        __pycache__/
        params_testDM.json
        ../OpenfMRI_ds000117_pybids/log.txt
        ../OpenfMRI_ds000117_pybids/params_nocli.json
        ../OpenfMRI_ds000117_pybids/prep_trans.py
        ../OpenfMRI_ds000117_pybids/trns.txt

aucune modification ajoutée à la validation mais des fichiers non suivis sont présents (utilisez "git add" pour les suivre)
```

This will modify the version of the file on the computer IRREVERSIBLY !!!!!!

Git

I made a mistake:

```
$ git stash
```

You erase all the modications that have NOT been commited so far (i.e. you go back to the latest pulled version)

```
$ git restore my/file.py
```

It is also possible to « restore » one single file instead of reloading all the modified files

This will modify the version of the file on the computer IRREVERSIBLY !!!!!!

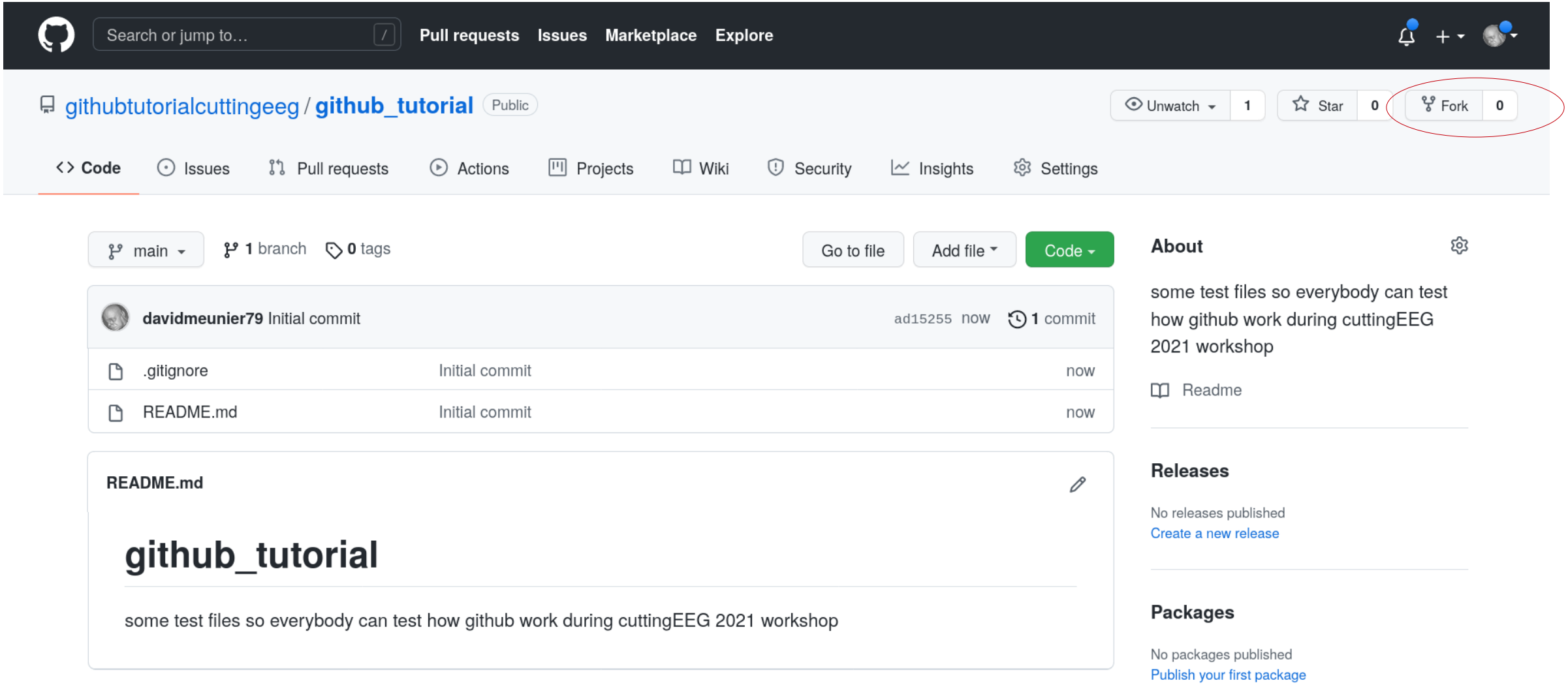
Git on Github

Git on Github

Advanced operations:

A better way of working :

- ***fork*** a version of the original project (remote = *upstream*) on your (github or else) account


Search or jump to…
Pull requests
Issues
Marketplace
Explore
githubtutorialcuttingeeg / github_tutorial
Public
Unwatch 1
Star 0
Fork 0
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
main
1 branch
0 tags
Go to file
Add file
Code
davidmeunier79 Initial commit
ad15255 now
1 commit
.gitignore
Initial commit
now
README.md
Initial commit
now
README.md
github_tutorial
some test files so everybody can test how github work during cuttingEEG 2021 workshop
About
some test files so everybody can test how github work during cuttingEEG 2021 workshop
Readme
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package

davidmeunier79 / **github_tutorial** Public

forked from githubtutorialcuttingeeg/github_tutorial

Watch 0 | Star 0 | Fork 1

Code | Pull requests | Actions | Projects | Wiki | Security | Insights | Settings

main | 1 branch | 0 tags | Go to file | Add file | Code

This branch is even with githubtutorialcuttingeeg:main. | Contribute | Fetch upstream

davidmeunier79 Initial commit ad15255 1 minute ago 1 commit

.gitignore	Initial commit	1 minute ago
README.md	Initial commit	1 minute ago

README.md

github_tutorial

some test files so everybody can test how github work during cuttingEEG 2021 workshop

About

some test files so everybody can test how github work during cuttingEEG 2021 workshop

Readme

Releases

No releases published

Create a new release

Packages

No packages published

Publish your first package

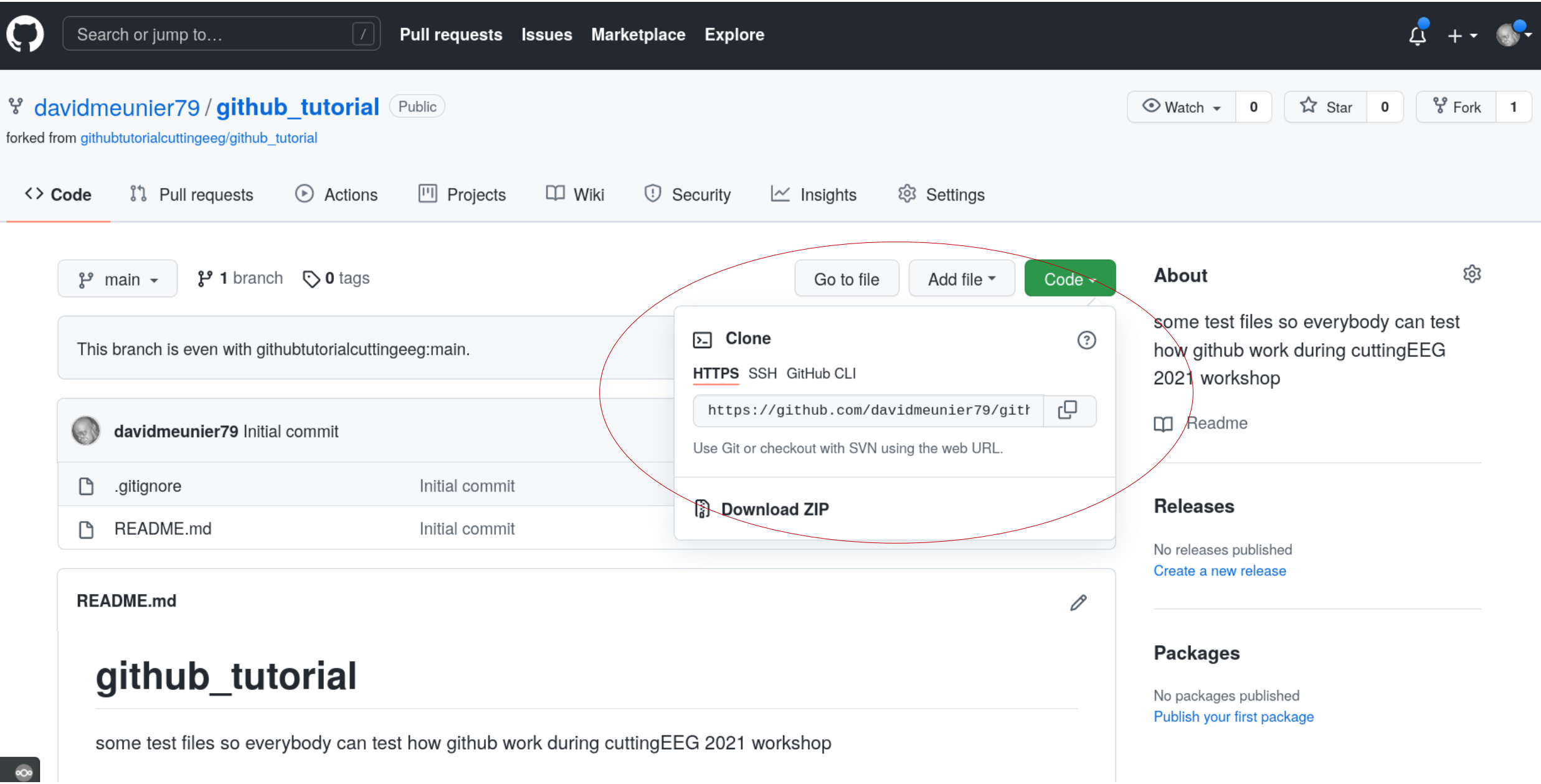
Git on Github

Advanced operations:

A better way of working :

- ***fork*** a version of the original project (remote = *upstream*) on your (github or else) account

- **clone** your local version from your forked remote (now named *origin*)


Search or jump to…
Pull requests
Issues
Marketplace
Explore
davidmeunier79 / github_tutorial
Public
forked from githubtutorialcuttingeeg/github_tutorial
Watch 0
Star 0
Fork 1
Code
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
main
1 branch
0 tags
Go to file
Add file
Code
This branch is even with githubtutorialcuttingeeg:main.
Clone
HTTPS SSH GitHub CLI
https://github.com/davidmeunier79/gith
Use Git or checkout with SVN using the web URL.
Download ZIP
davidmeunier79 Initial commit
.gitignore Initial commit
README.md Initial commit
About
some test files so everybody can test how github work during cuttingEEG 2021 workshop
Readme
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package
README.md
github_tutorial
some test files so everybody can test how github work during cuttingEEG 2021 workshop

Git on Github

Advanced operations:

A better way of working :

- ***fork*** a version of the original project (remote = *upstream*) on your (github or else) account

- **clone** your local version from your forked remote (now named *origin*)

```
$ git clone my_repo/my_project
```

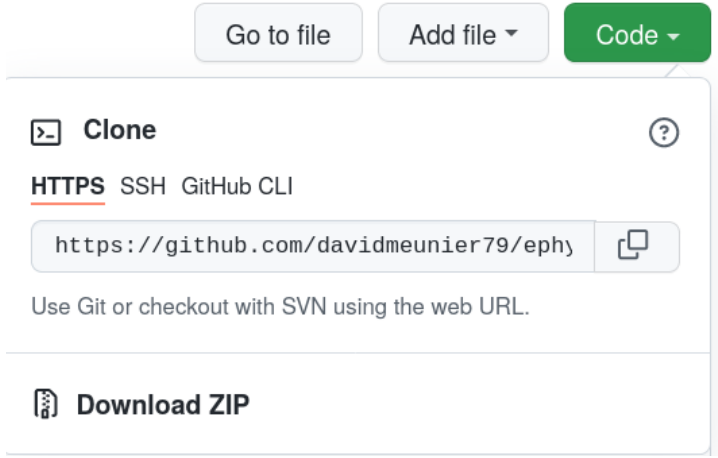
Git on Github

Advanced operations:

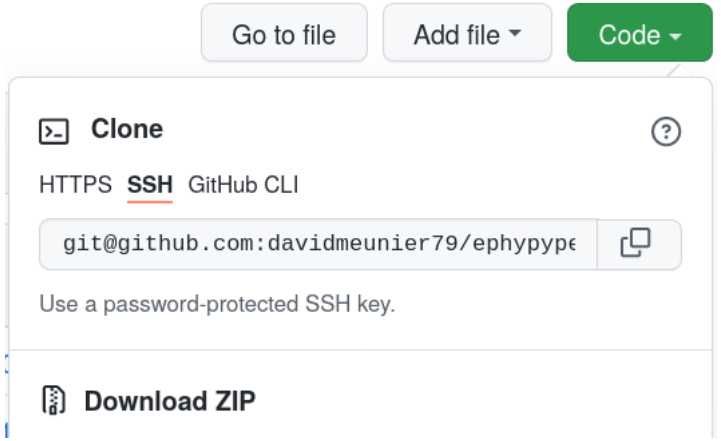
A better way of working :

- ***fork*** a version of the original project (remote = *upstream*) on your (github or else) account
- **clone** your local version from your forked remote (now named *origin*)

Side note : be careful, on github since a few months :



https requires 2 Factor Authentification 2-FA to push, but is ok if you clone only



ssh with public/private key is still OK

Git on Github

Advanced operations:

A better way of working :

- ***fork*** a version of the original project (remote = *upstream*) on your (github or else) account

- **clone** your local version from your forked remote (now named *origin*)

- create a new **branch** (*my_new_feature*)

```
$ git branch my_new_feature
$ git checkout my_new_feature
```

Git on Github

Advanced operations:

A better way of working :

- ***fork*** a version of the original project (remote = *upstream*) on your (github or else) account
- **clone** your local version from your forked remote (now named *origin*)
- create a new **branch** (*my_new_feature*)
- work on your local version, and make commit/push on *my_new_feature*

Git on Github

Advanced operations:

A better way of working :

- ***fork*** a version of the original project (remote = *upstream*) on your (github or else) account

- **clone** your local version from your forked remote (now named *origin*)

- create a new **branch** (*my_new_feature*)

- work on your local version, and make commit/push on *my_new_feature*

- propose a **Pull Request (PR)** when a « real » improvement have been added to the project.

This step involves both automatic and « human » evaluation of the quality of the code (formatting, naming, etc) and checks if the modification do not disrupt the project as a whole **(**called **continuous integration)**.

Git on Github

Advanced operations:

A better way of working :

- ***fork*** a version of the original project (remote = *upstream*) on your (github or else) account

- **clone** your local version from your forked remote (now named *origin*)

- create a new **branch** (*my_new_feature*)

- work on your local version, and make commit/push on *my_new_feature*

- propose a **Pull Request (PR)** when a « real » improvement have been added to the project.

This step involves both automatic and « human » evaluation of the quality of the code (formatting, naming, etc) and checks if the modification do not disrupt the project as a whole **(**called **continuous integration)**.

A good practice is to **retrieve** the last version of the original project before :

```
$ git remote add upstream https://github.com/neuropycon/ephypype.git
$ git pull upstream master
```

Pull Request on github

[MRG] Hotfix #47

Open **davidmeunier79** wants to merge 3 commits into `neuropycon:master` from `davidmeunier79:hotfix`

Conversation 1 | Commits 3 | Checks 0 | Files changed 15

davidmeunier79 commented 10 days ago — Member

No description provided.

`started example with dyn graph, ok till series of conf cormat` `99ef953`

davidmeunier79 force-pushed the `davidmeunier79:hotfix` branch from **`19f834a`** to **`bfbcd76`** 10 days ago

Pull Request on github

[MRG] Hotfix #47

Open **davidmeunier79** wants to merge 3 commits into `neuropycon:master` from `davidmeunier79:hotfix`

Conversation 1 | Commits 3 | Checks 0 | Files changed 15

davidmeunier79 commented 10 days ago — Member

No description provided.

`started example with dyn graph, ok till series of conf cormat` `99ef953`

davidmeunier79 force-pushed the `davidmeunier79:hotfix` branch from **`19f834a`** to **`bfbcd76`** 10 days ago

Pull Request on github

davidmeunier79 added 3 commits 18 hours ago

use priors and no mesh in default params	c765dc2
segkmeans	364eebd
test if false	a2774f8

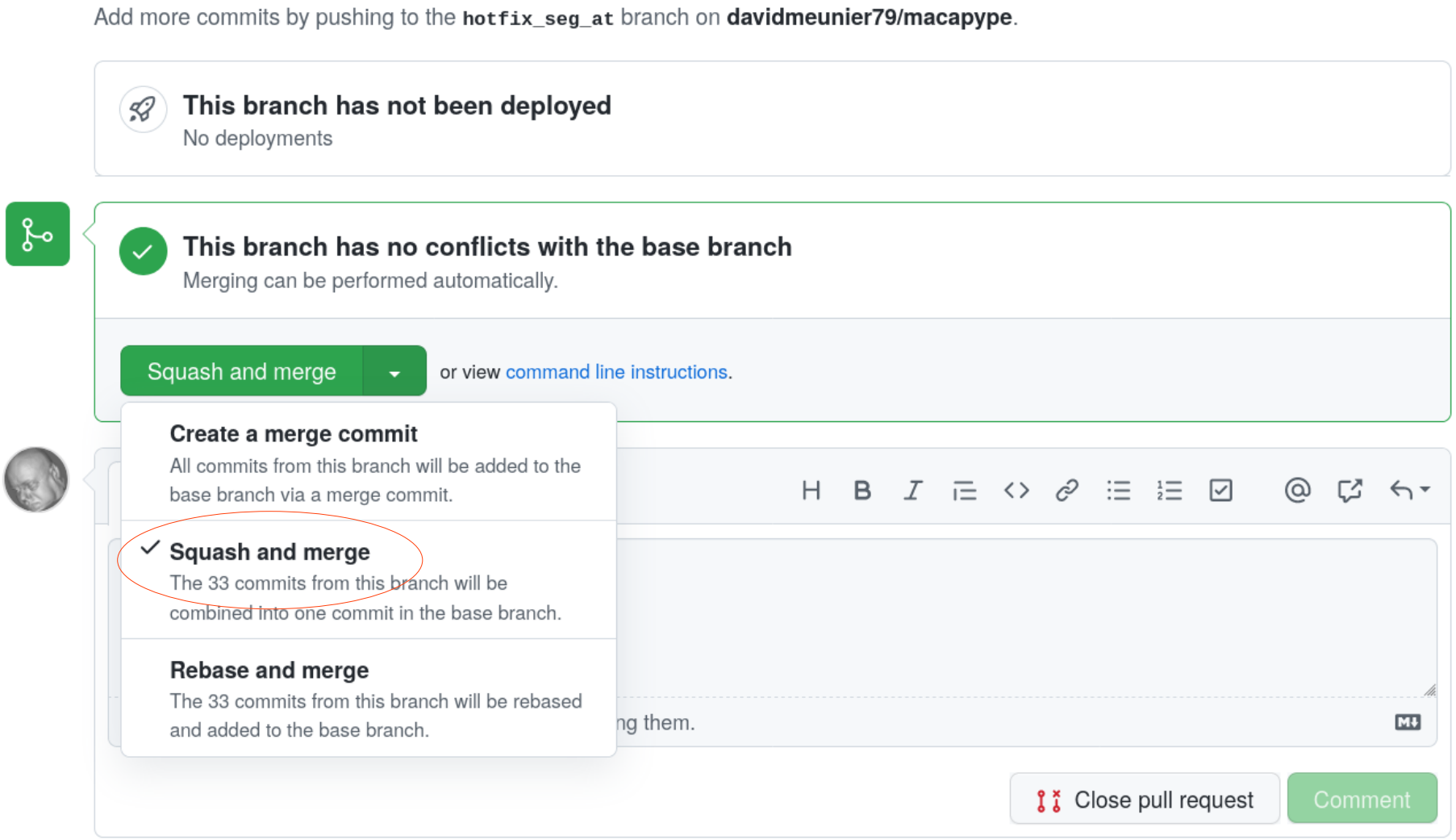
Add more commits by pushing to the **`hotfix_seg_at`** branch on **davidmeunier79/macapype**.

This branch has not been deployed
No deployments

This branch has no conflicts with the base branch
Merging can be performed automatically.

Squash and merge or view command line instructions.

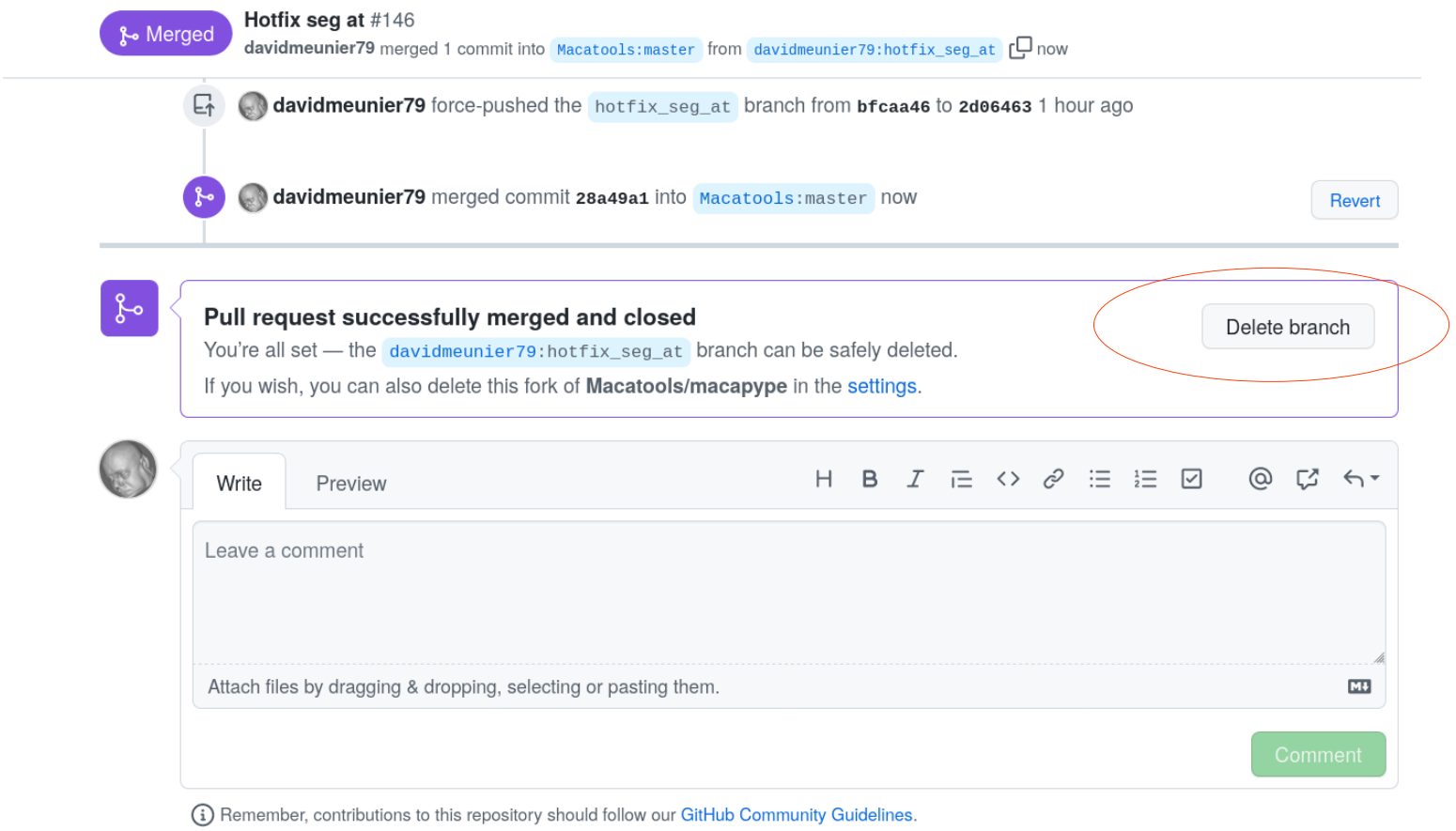
Pull Request on github



After a Pull Request

Once a PR is accepted, the good practice is to :

- delete the corresponding branch on your github fork (eg davidmeunier79:my_feature)



After a Pull Request

Once a PR is accepted, the good practice is to :

- delete the corresponding branch on your github fork (eg davidmeunier79:my_feature)

- on your computer, in your local version :

```
$ git checkout master # get back to the master/main branch
$ git pull upstream master  # get the master verion updated with the « new » clean version of the package
$ git branch -D my_feature # DELETE the branch you were working on
```

Thanks for your attention !