Git training

Nicolas Barrier   Witold Podlejski
Criscely Lujan-Paredes

Presentation of the Git Software

What is Git?

- Free and open source software
- Light and local use (without internet)
- The most popular Version Control Software (VCS)
- Manages and tracks versions of a project (code, manuscript, data)
- Can be linked with remote server (GitHub, Gitlab)

What is Git for?

- For a single user:
  - **Track changes** (*commits*) over time with information about **when** and **what** are the changes
  - Eventually go **back in time**

  - **Synchronize** the project in the **cloud** with git servers (GitHub, Gitlab)

What is Git for ?

- For a collaborative project:
  - **Track changes** (*commits*) with information about **who**, **when** and **what** are the changes
  - **Resolve version conflict** when simultaneous changes
  - **Highlight a specific version** of the project (*tags*)
  - New version of a software
  - Submitted, revised versions of a paper
  - Create **derivates** of a project (*branches*)
    - Production
    - Development
    - Feature
  - **Publish** the project (open science)

In short...

Installation and configurations

Installing Git

Windows and Mac

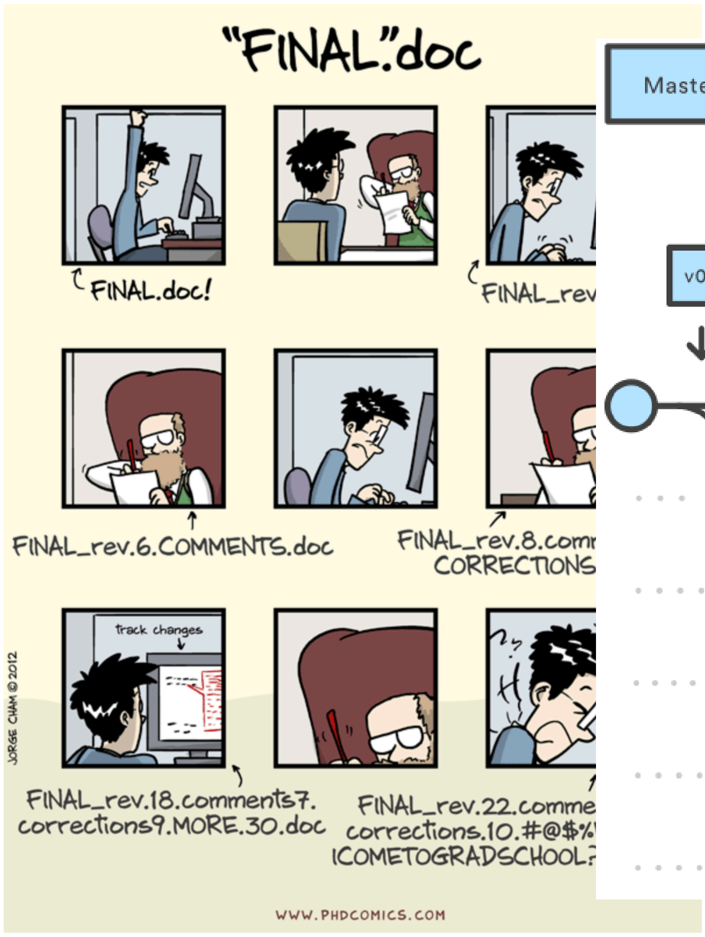


Figure 1: Without Git

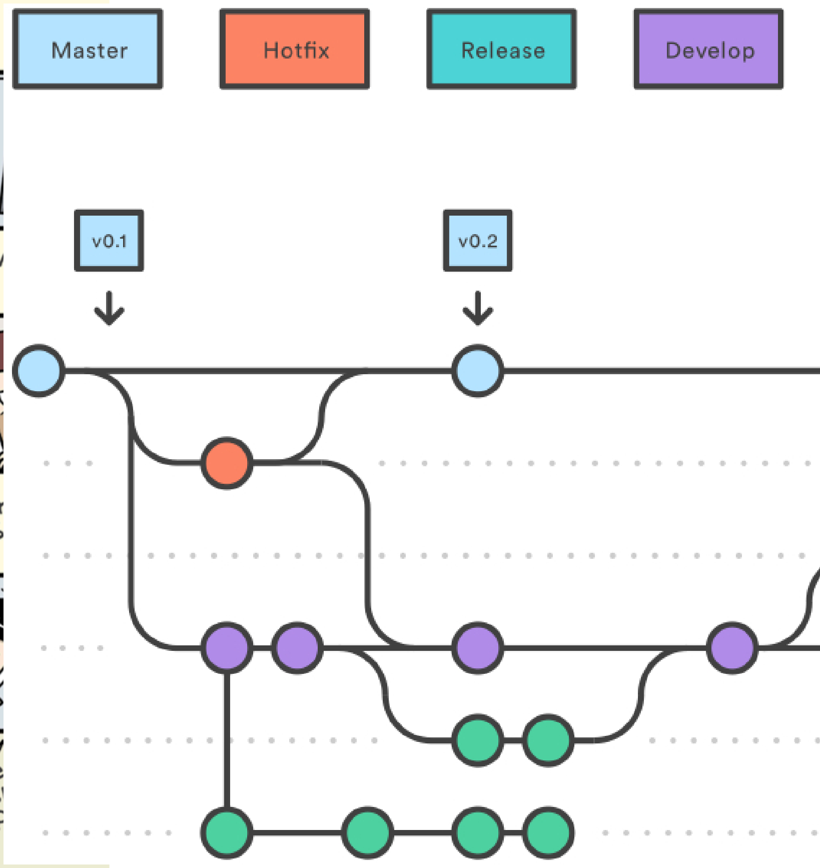


Figure 2: With Git

Download and install Git from https://git-scm.com/downloads.

When done, open `Git Bash`

Linux

Open a `Terminal` window and type:

```
sudo apt install git git-lfs git-flow
```

Git configuration

On `Git Bash` or in the `Terminal`:

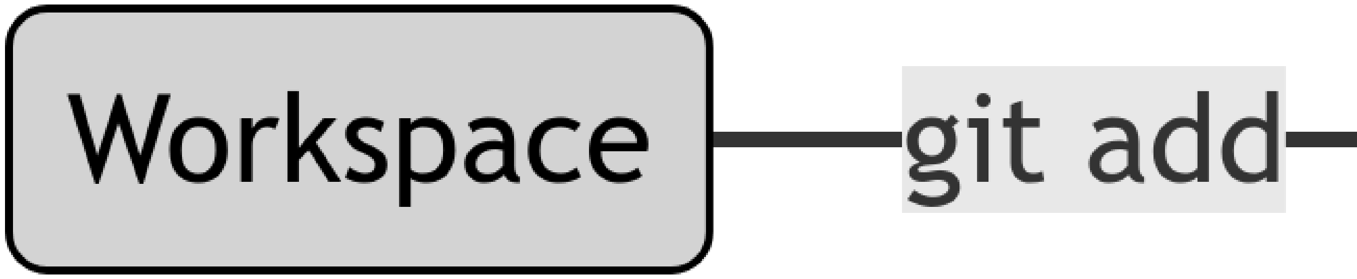
- Type `git config --global user.name "Firstname Lastname"`
- Type `git config --global user.email "myadresse@ird.fr"`

> **Note**
>
> These two lines identify you in the history of a project.

- Type `git config --global --list` to see the global git configuration.

Getting started with Git in local

Git architecture



- `Workspace`: your working directory → your computer
- `Local`: the local repository → contains the history of your project
- `Index`: a buffer between `Workspace` and `Local` → list of the files that will be sent from `Workspace` to `Local`
- `git add` : the command to add the file(s) in the `Index`
- `git commit`: the command to validate the changes (moves the files from `Index` to `Local`)

Getting started

- Create a folder called `training-git` by typing `mkdir training-git`
- Move to the folder by typing `cd training-git`
- Type `ls -alrt`
- Type `git init`
- Type again `ls -alrt`.

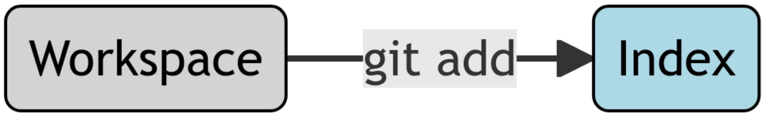
Note

A `.git` folder has appeared. It contains the full history of your project (`Local` repository)

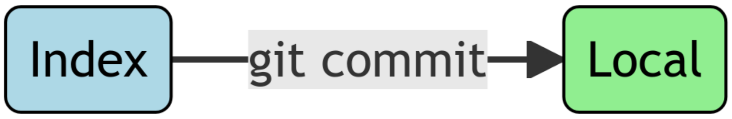
- Type `git status` and `git log`

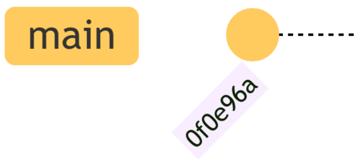
First commit

- Create a `README.md` file. Type `git status` → `README.md` is now in `Workspace` but not in `Index` nor in `Local`
- Type `git add README.md` and `git status`



- Type `git commit -m "First commit"` and type `git log`



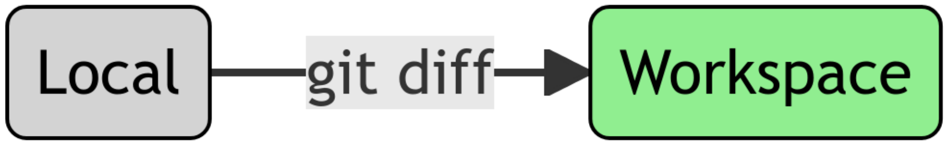


Note

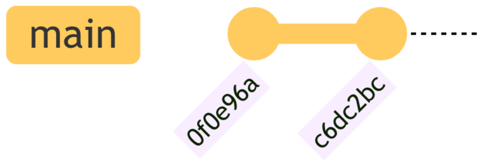
`0f0e96a` is a short version of the identifier of the commit

Second commit

- Open the `README.md` file, add `# Git training` and save
- Type `git status`
- Type `git diff`

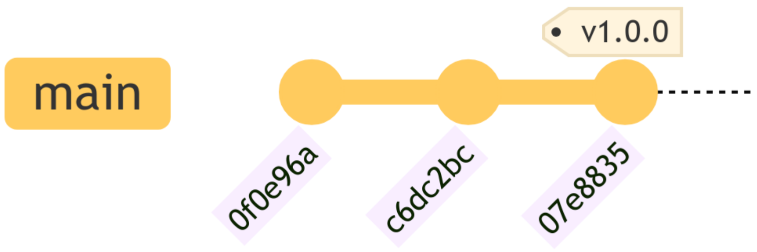


- Type `git commit -m "Second commit"`
- Type `git log`



Creating tags

- Open the `README.md` file and add `## Version v1.0.0`.
- Type `git add README.md`
- Type `git commit -m "Third commit"`
- Type `git tag v1.0.0` and `git log`



- Type `git tag` to list all existing tags

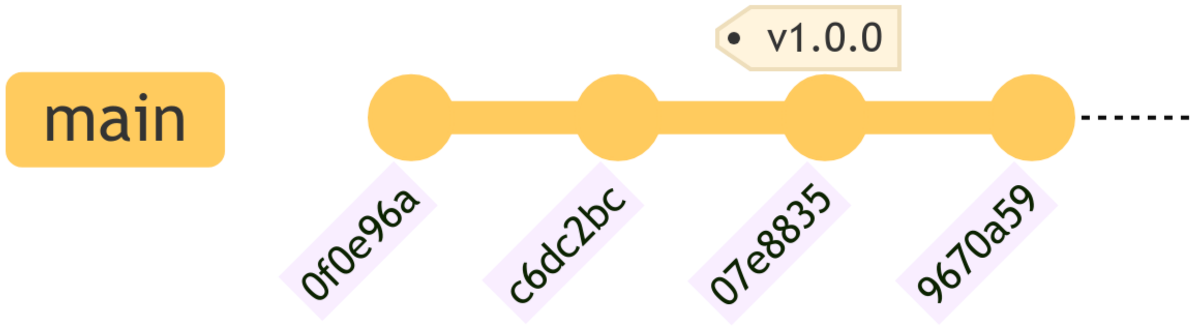
Ignoring files

It is possible to tell Git to ignore some files by using a .gitignore file.

- Create an empty `output.log` file and type `git status`
- Create a `.gitignore` file and write `*.log`. Type again `git status`

The `output.log` file does not appear as an `Untracked file` anymore

- Type `git add .gitignore` and `git status`
- Type `git commit -m "Fourth commit"`

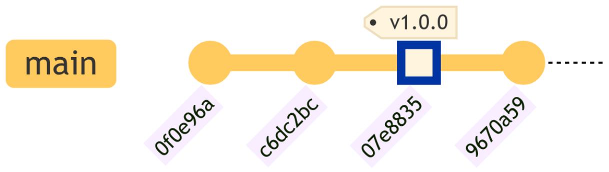


Tip

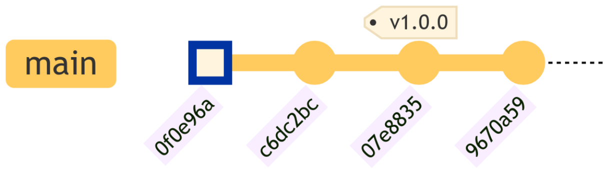
To list the ignored files, type `git ls-files --others --ignored --exclude-from=.gitignore`

Moving in the history

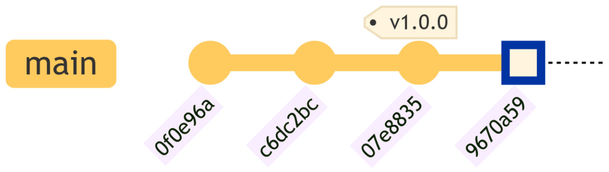
- Type `git checkout v1.0.0` → move to a tag



- Type `git checkout 0f0e96a` → move to a specific commit



- Type `git checkout main` → move at the latest commit (replace `main` by `master` if the latter is the name of the main branch)



Tip

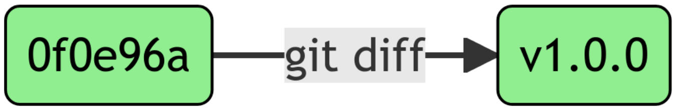
`HEAD` is a symbolic reference pointing to wherever you are in your commit history, as shown in `git log`

Display differences

- Type `git diff 0f0e96a v1.0.0` → compares a commit and a tag.

Warning

Order matters when using `git diff`. Differences are shown with the reference state considered to be the first argument.



- Type `git diff 0f0e96a c6dc2bc` → compares two commits.
- Type `git diff 0f0e96a HEAD` → compares where you are in the history (`HEAD`) with a given commit.

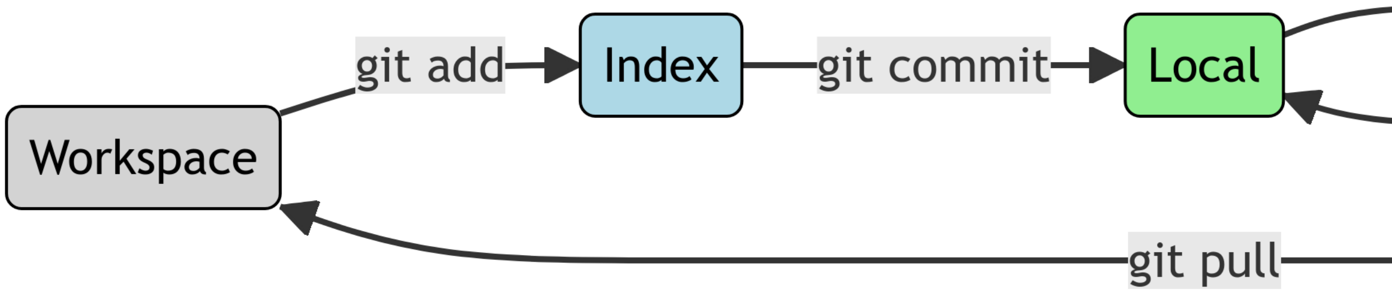
Using Git with online server (GitHub)

Using remotes

In addition of saving the history, Git has other advantages. It allows to:

- Save a project remotely
  - Synchronization with different computers (laptop, HPCs)
- Share a project (codes, packages) with the community
  - Reproducible science

To do so, a 4th component in the Git architecture must be considered: the `Remote` repository. It contains a **remote** version of the history of your project



Remote hosts

There are several remote hosting possibilities:

Commercial hosts:

- GitHub: https://github.com/
- GitLab: https://gitlab.com/

Institutional hosts:

- GitLab IRD: https://forge.ird.fr/
- GitLab Ifremer https://gitlab.ifremer.fr/

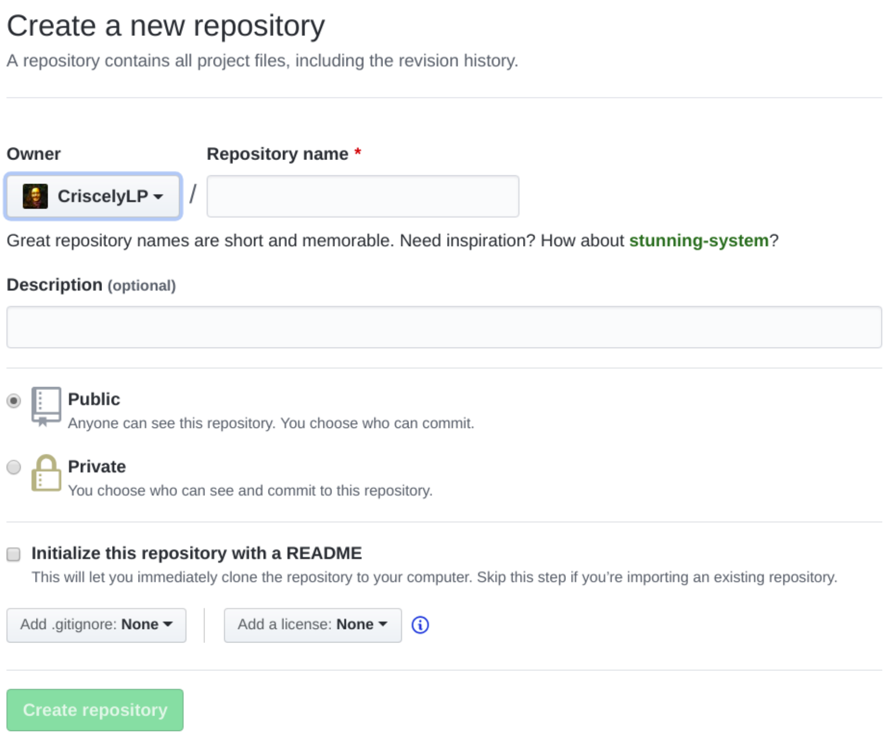
In the following, we will use GitHub.

Tip

GitHub proposes extra-features for students, teachers and researchers. Visit https://education.github.com/benefits for more informations

Creation of a GitHub repository

- On your GitHub page, click on `Repositories`
- Click on the the green `New` button
- Set the name of your remote repository. Leave the other fields empty
- Click on `Create repository`



Create a new repository

A repository contains all project files, including the revision history.

Owner: CriscelyLP / Repository name *

Great repository names are short and memorable. Need inspiration? How about **stunning-system**?

Description (optional)

● Public
Anyone can see this repository. You choose who can commit.

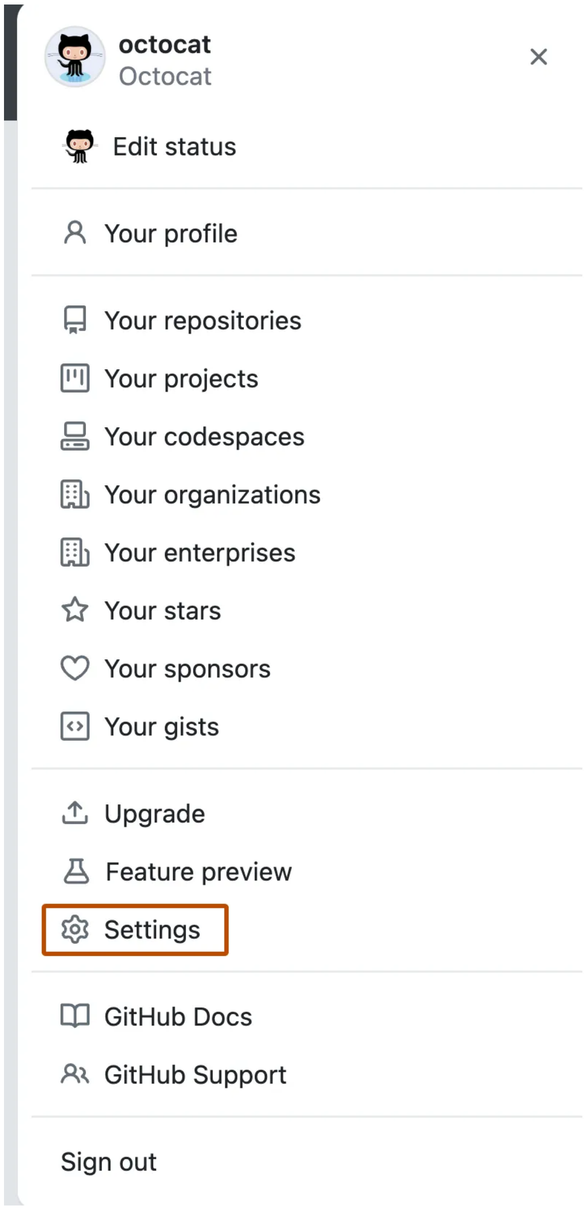
○ Private
You choose who can see and commit to this repository.

☐ Initialize this repository with a README
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Creation of a personal access token

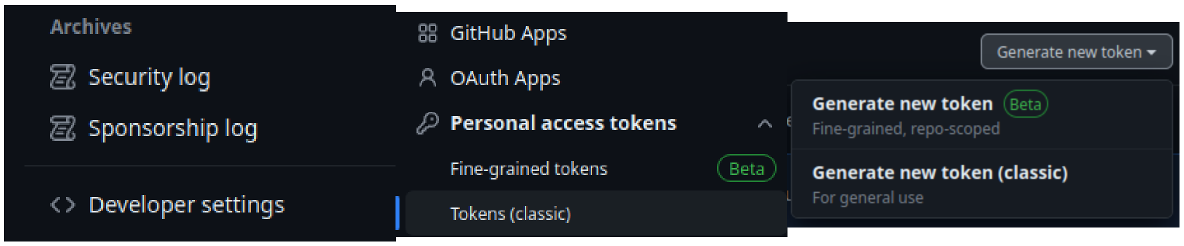
To authentificate, you need to create an authentification token (see GitHub authentification of details for details).

To do so, click on your profile photo and then on `Settings`:



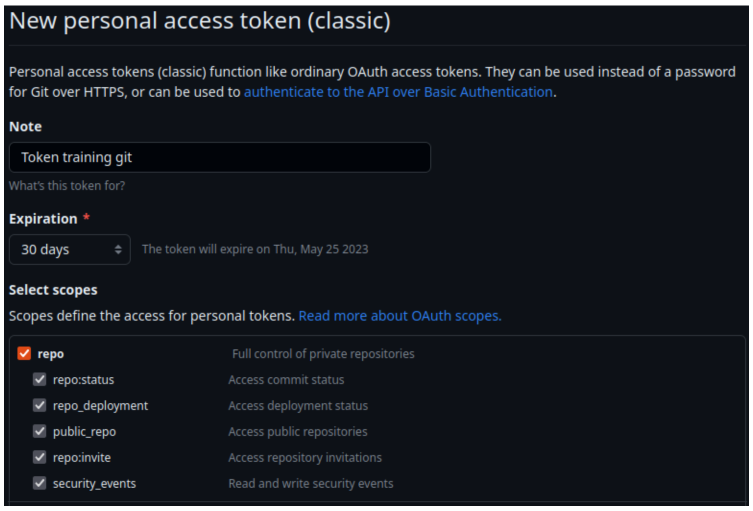
Creation of a personal access token

- In the left sidebar, click on `Developer settings`.
- Under `Personal access tokens`, click `Tokens (classic)`.
- Select `Generate new token` and `Generate new token (classic)`.



Creation of a personal access token

- Add a description note and **click on the "repo" box**, as shown below:


New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.

Note

Token training git

What's this token for?

Expiration *

30 days — The token will expire on Thu, May 25 2023

Select scopes

Scopes define the access for personal tokens. Read more about OAuth scopes.

✓ repo	Full control of private repositories
✓ repo:status	Access commit status
✓ repo_deployment	Access deployment status
✓ public_repo	Access public repositories
✓ repo:invite	Access repository invitations
✓ security_events	Read and write security events

- Click on the `Generate token` box button.
- **Copy and save in a `.txt` file or in a Password manager tool (KeePassXC) the token: this is your password!** It should look like something like this:

```
ghp_****************************************
```

Linking Git local and remote

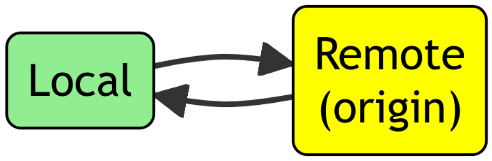
- In `Terminal` or `Git Bash`, type the following line:

```
git remote add origin https://github.com/barriern/git-train.git
```

⚠ Warning

Replac `barriern` by your GitHub login and `git-train` by the name of your GitHub repository.

- It connects your Local repository with a remote one, called *origin*



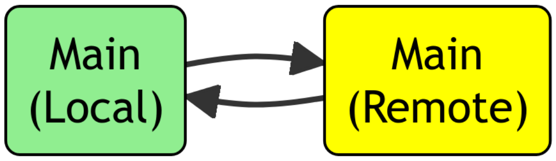
- Type `git remote -vv`

Linking Git local and remote

Now that the local and remote repositories are linked, the same thing must be done with the branches.

- Type `git branch -M main` by replacing `main` by the name of the remote branch on GitHub. It will rename the local branch with the same name.
- Type `git push -u origin main`

It connects the *local* and *remote* main branches (`-u` option) and sends the commits to the remote branch

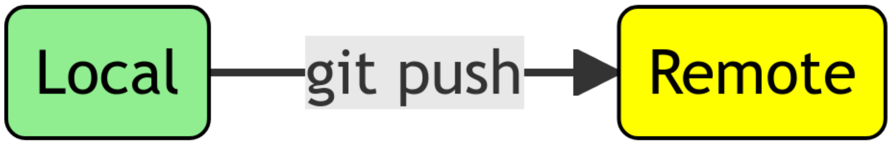


- Type `git branch -vv`

Linking Git local and remote

Have a look at your repository on GitHub. **Tags are missing!**

Type `git push --tags` and refresh the GitHub page.



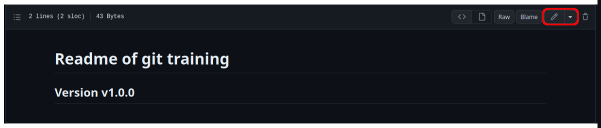
Note

No need to specify the `-u origin main` arguments since the two branches are already connected.

Navigate on the GitHub page to see what has been done.

Synchronization from the remote

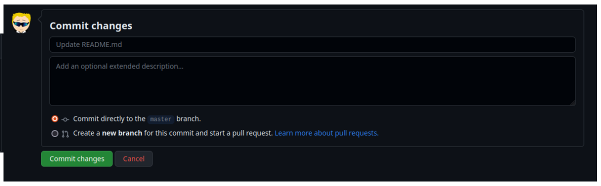
- In GitHub, click on the `README.md` file and then on the edit button
- Add a `Update from Github` line and click on `Commit changes`



2 lines (2 sloc) | 43 Bytes

Readme of git training

Version v1.0.0



Commit changes

Update README.md

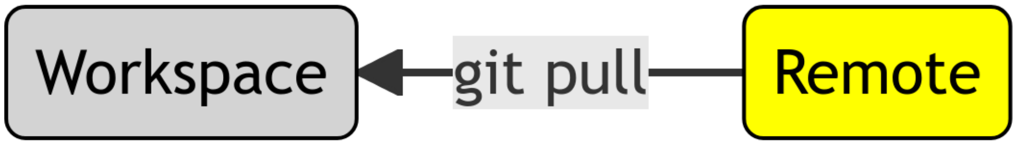
Add an optional extended description...

[✓ Commit directly to the `master` branch.]

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

The `Remote` change of `README.md` is not yet visible in `Workspace`!

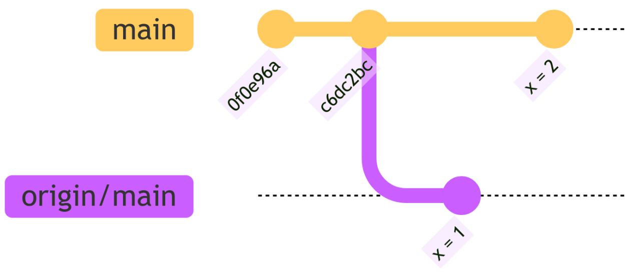
- In `Git Bash` or `Terminal`, type `git pull`



- Look again at the `README.md` file on your computer. You should see the update.

Synchronization: conflicts

- On GitHub, add `x = 1` at the end of the `README.md` file. **Do not type pull!**
- On your computer, edit the `README.md` and add `x = 2`.
- Type `git add README.md`
- Type `git commit -m "Fifth commit"`
- Type `git push`. An error occurs because changes in `Remote` have not been pulled in `Local`.
- Type `git pull` and `git status`. An error occurs because there is a conflict in the `README.md` file which cannot be solved by Git.



Synchronization: conflicts

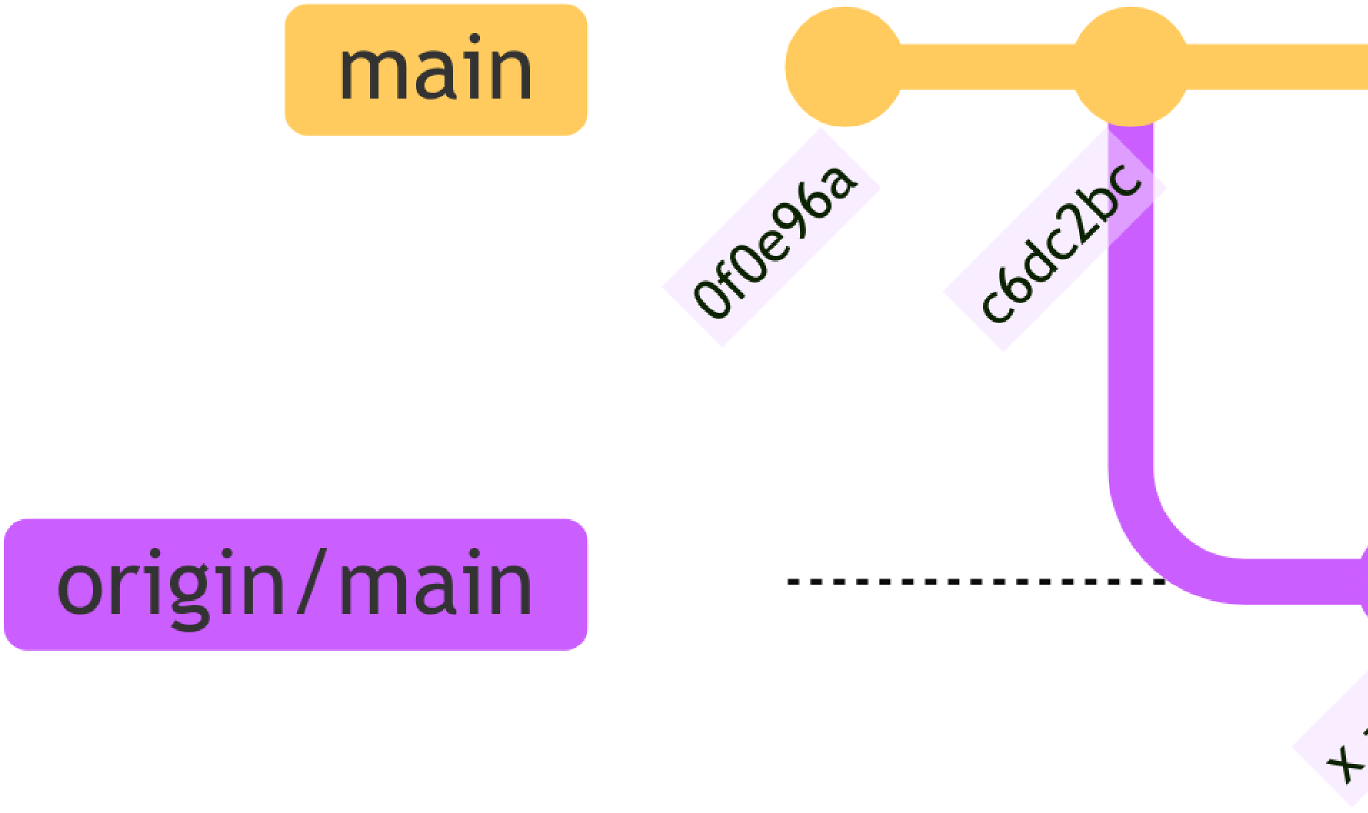
- Open the `README.md` file. You should see:

```
<<<<<<< HEAD
x = 2
=======
x = 1
>>>>>>> 70a4c105e377db282c0769606960f0afcccdd071
```

Warning

These are conflicts markers. `Git` doesn't know whether to chose `x = 1` or `x = 2`. This is your job

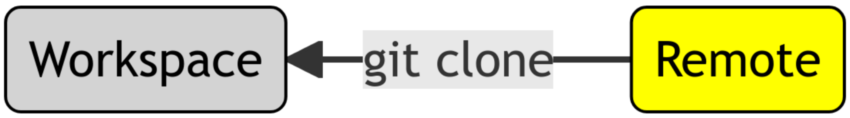
- Open the file, replace the above by `x = 3`. Commit and push the changes



Cloning an existing repository

- In `Terminal` or `Git Bash`, type `cd ..`

- Now type `git clone https://github.com/umr-marbec/git-training`



- Type `git log` to see the full history.
- To update the project, type `git pull`

> ⚠ Warning
>
> **Do not clone or initialize a Git repository in another Git repository!**

Create a repository the simple way

To create a new repo more simply than done here:

- Create a repo. on GitHub with a `README.md` file and eventually a `LICENCE` file.
- Clone the repo.
- You are all set!
  - The `remote` and `local` repositories are synchonized
  - The `remote` and `local` main branches are synchonized

Conclusion: good practice

- Before starting editing a project, do a `git pull`
- Commit very often using `git commit` extensively

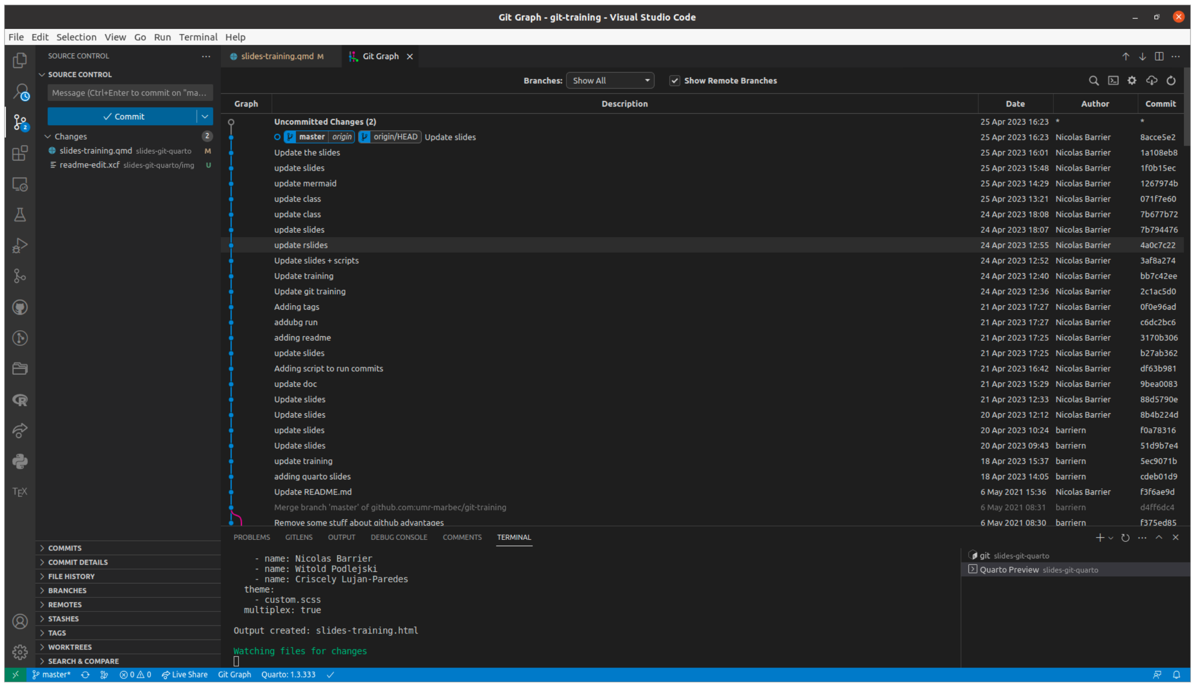
- Push often as well using `git push`
- Use `git status` extensively to know what you have done

Git clients

Git clients: what is it?

Git Clients are softwares that facilitate the use of Git (see Git Guis for a list).

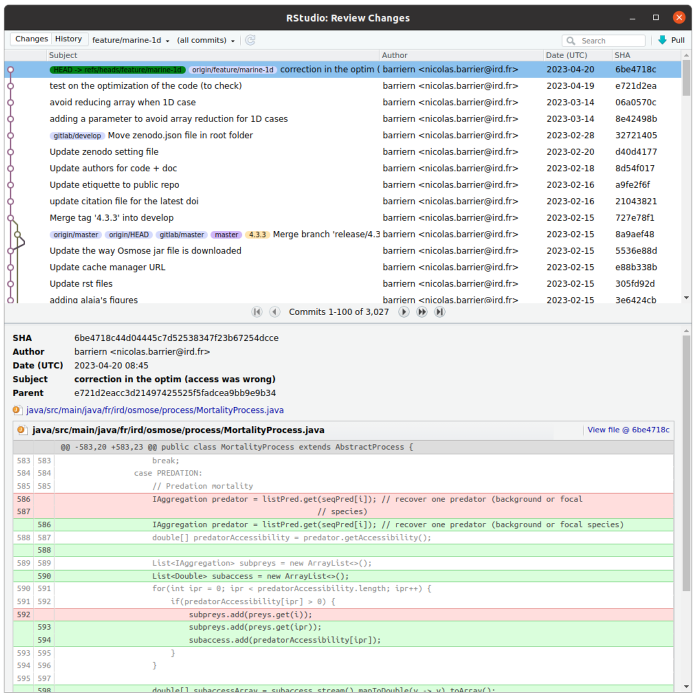
Beside, most code editors include Git functionalities


Git Graph - git-training - Visual Studio Code
File Edit Selection View Go Run Terminal Help
SOURCE CONTROL
Message (Ctrl+Enter to commit on "ma...
✓ Commit
Changes
slides-training.qmd slides-git-quarto M
readme-edit.xcf slides-git-quarto/img U
slides-training.qmd M
Git Graph
Branches: Show All
Show Remote Branches
COMMITS
COMMIT DETAILS
FILE HISTORY
BRANCHES
REMOTES
STASHES
TAGS
WORKTREES
SEARCH & COMPARE
PROBLEMS GITLENS OUTPUT DEBUG CONSOLE COMMENTS TERMINAL
git slides-git-quarto
Quarto Preview slides-git-quarto
master* Live Share Git Graph Quarto: 1.3.333

Graph	Description	Date	Author	Commit
	Uncommitted Changes (2)	25 Apr 2023 16:23	*	*
	master origin origin/HEAD Update slides	25 Apr 2023 16:23	Nicolas Barrier	8acce5e2
	Update the slides	25 Apr 2023 16:01	Nicolas Barrier	1a108eb8
	update slides	25 Apr 2023 15:48	Nicolas Barrier	1f0b15ec
	update mermaid	25 Apr 2023 14:29	Nicolas Barrier	1267974b
	update class	25 Apr 2023 13:21	Nicolas Barrier	071f7e60
	update class	24 Apr 2023 18:08	Nicolas Barrier	7b677b72
	update slides	24 Apr 2023 18:07	Nicolas Barrier	7b794476
	update rslides	24 Apr 2023 12:55	Nicolas Barrier	4a0c7c22
	Update slides + scripts	24 Apr 2023 12:52	Nicolas Barrier	3af8a274
	Update training	24 Apr 2023 12:40	Nicolas Barrier	bb7c42ee
	Update git training	24 Apr 2023 12:36	Nicolas Barrier	2c1ac5d0
	Adding tags	21 Apr 2023 17:27	Nicolas Barrier	0f0e96ad
	addubg run	21 Apr 2023 17:27	Nicolas Barrier	c6dc2bc6
	adding readme	21 Apr 2023 17:25	Nicolas Barrier	3170b306
	update slides	21 Apr 2023 17:25	Nicolas Barrier	b27ab362
	Adding script to run commits	21 Apr 2023 16:42	Nicolas Barrier	df63b981
	update doc	21 Apr 2023 15:29	Nicolas Barrier	9bea0083
	Update slides	21 Apr 2023 12:33	Nicolas Barrier	88d5790e
	Update slides	20 Apr 2023 12:12	Nicolas Barrier	8b4b224d
	update slides	20 Apr 2023 10:24	barriern	f0a78316
	Update slides	20 Apr 2023 09:43	barriern	51d9b7e4
	update training	18 Apr 2023 15:37	barriern	5ec9071b
	adding quarto slides	18 Apr 2023 14:05	barriern	cdeb01d9
	Update README.md	6 May 2021 15:36	Nicolas Barrier	f3f6ae9d
	Merge branch 'master' of github.com:umr-marbec/git-training	6 May 2021 08:31	barriern	d4ff6dc4
	Remove some stuff about github advantages	6 May 2021 08:30	barriern	f375ed85

```
    - name: Nicolas Barrier
    - name: Witold Podlejski
    - name: Criscely Lujan-Paredes
  theme:
    - custom.scss
  multiplex: true

Output created: slides-training.html

Watching files for changes
```

Figure 3: VSCode



	Subject	Author	Date (UTC)	SHA
	HEAD -> refs/heads/feature/marine-1d origin/feature/marine-1d correction in the optim (	barriern <nicolas.barrier@ird.fr>	2023-04-20	6be4718c
	test on the optimization of the code (to check)	barriern <nicolas.barrier@ird.fr>	2023-04-19	e721d2ea
	avoid reducing array when 1D case	barriern <nicolas.barrier@ird.fr>	2023-03-14	06a0570c
	adding a parameter to avoid array reduction for 1D cases	barriern <nicolas.barrier@ird.fr>	2023-03-14	8e42498b
	gitlab/develop Move zenodo.json file in root folder	barriern <nicolas.barrier@ird.fr>	2023-02-28	32721405
	Update zenodo setting file	barriern <nicolas.barrier@ird.fr>	2023-02-20	d40d4177
	Update authors for code + doc	barriern <nicolas.barrier@ird.fr>	2023-02-18	8d54f017
	Update etiquette to public repo	barriern <nicolas.barrier@ird.fr>	2023-02-16	a9fe2f6f
	update citation file for the latest doi	barriern <nicolas.barrier@ird.fr>	2023-02-16	21043821
	Merge tag '4.3.3' into develop	barriern <nicolas.barrier@ird.fr>	2023-02-15	727e78f1
	origin/master origin/HEAD gitlab/master master 4.3.3 Merge branch 'release/4.3	barriern <nicolas.barrier@ird.fr>	2023-02-15	8a9aef48
	Update the way Osmose jar file is downloaded	barriern <nicolas.barrier@ird.fr>	2023-02-15	5536e88d
	Update cache manager URL	barriern <nicolas.barrier@ird.fr>	2023-02-15	e88b338b
	Update rst files	barriern <nicolas.barrier@ird.fr>	2023-02-15	305fd92d
	adding alaia's figures	barriern <nicolas.barrier@ird.fr>	2023-02-15	3e6424cb

Commits 1-100 of 3,027

SHA 6be4718c44d04445c7d52538347f23b67254dcce
Author barriern <nicolas.barrier@ird.fr>
Date (UTC) 2023-04-20 08:45
Subject **correction in the optim (access was wrong)**
Parent e721d2eacc3d21497425525f5fadcea9bb9e9b34

java/src/main/java/fr/ird/osmose/process/MortalityProcess.java

java/src/main/java/fr/ird/osmose/process/MortalityProcess.java

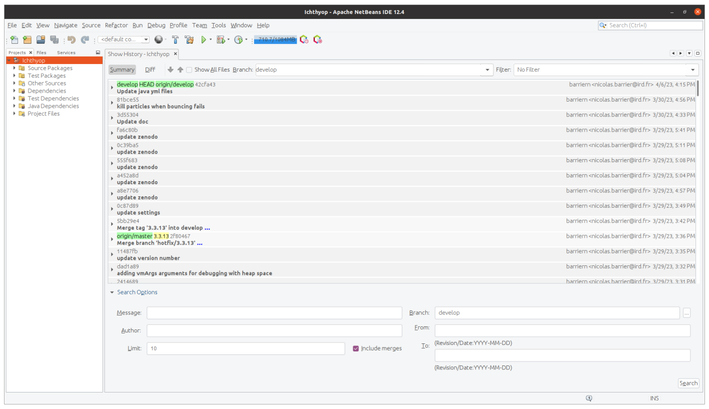
```
@@ -583,20 +583,23 @@ public class MortalityProcess extends AbstractProcess {
                    break;
                case PREDATION:
                    // Predation mortality
                    IAggregation predator = listPred.get(seqPred[i]); // recover one predator (background or focal
                                                                      // species)
                    IAggregation predator = listPred.get(seqPred[i]); // recover one predator (background or focal species)
                    double[] predatorAccessibility = predator.getAccessibility();

                    List<IAggregation> subpreys = new ArrayList<>();
                    List<Double> subaccess = new ArrayList<>();
                    for(int ipr = 0; ipr < predatorAccessibility.length; ipr++) {
                        if(predatorAccessibility[ipr] > 0) {
                            subpreys.add(preys.get(i));
                            subpreys.add(preys.get(ipr));
                            subaccess.add(predatorAccessibility[ipr]);
                        }
                    }

                    double[] subaccessArray = subaccess.stream().mapToDouble(v -> v).toArray();
```

Figure 4: RStudio

Git clients

Git clients

Git clients



Show All Files Branch: develop Filter: No Filter

```
develop HEAD origin/develop 42cfa43                     barriern <nicolas.barrier@ird.fr> 4/6/23, 4:15 PM
Update java yml files
81bce55                                                 barriern <nicolas.barrier@ird.fr> 3/30/23, 4:56 PM
kill particles when bouncing fails
3d55304                                                 barriern <nicolas.barrier@ird.fr> 3/30/23, 4:33 PM
Update doc
fa6c80b                                                 barriern <nicolas.barrier@ird.fr> 3/29/23, 5:41 PM
update zenodo
0c39ba5                                                 barriern <nicolas.barrier@ird.fr> 3/29/23, 5:11 PM
update zenodo
555f683                                                 barriern <nicolas.barrier@ird.fr> 3/29/23, 5:08 PM
update zenodo
a452a8d                                                 barriern <nicolas.barrier@ird.fr> 3/29/23, 5:04 PM
update zenodo
a8e7706                                                 barriern <nicolas.barrier@ird.fr> 3/29/23, 4:57 PM
update zenodo
0c87d89                                                 barriern <nicolas.barrier@ird.fr> 3/29/23, 3:49 PM
update settings
5bb29e4                                                 barriern <nicolas.barrier@ird.fr> 3/29/23, 3:42 PM
Merge tag '3.3.13' into develop ...
origin/master 3.3.13 2f80467                            barriern <nicolas.barrier@ird.fr> 3/29/23, 3:36 PM
Merge branch 'hotfix/3.3.13' ...
11487fb                                                 barriern <nicolas.barrier@ird.fr> 3/29/23, 3:35 PM
update version number
dad1a89                                                 barriern <nicolas.barrier@ird.fr> 3/29/23, 3:32 PM
adding vmArgs arguments for debugging with heap space
2414689                                                 barriern <nicolas.barrier@ird.fr> 3/29/23, 3:31 PM
```

Message: Branch: develop
Author: From:
Limit: 10 ☑ Include merges (Revision/Date:YYYY-MM-DD)
To: (Revision/Date:YYYY-MM-DD)

INS

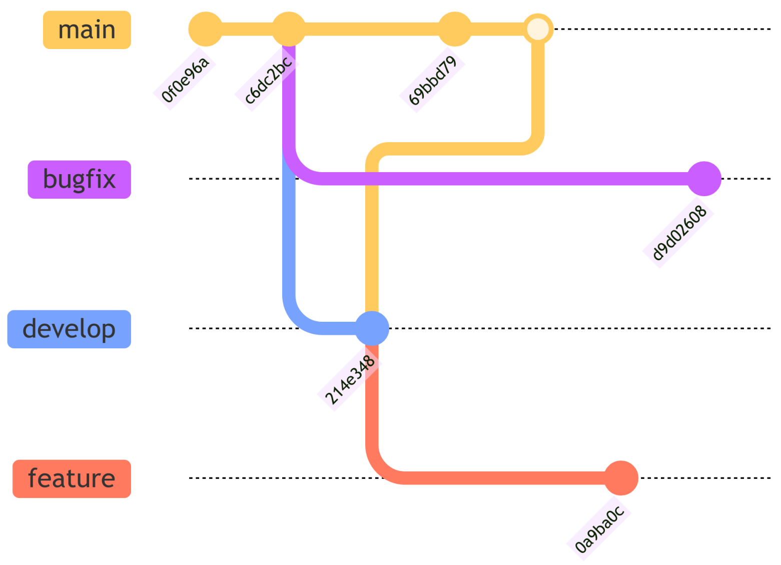
Figure 5: Netbeans

Going further

Going further...

For those who want, extra slides are available on:

- Git with Large File Storage extension.
- Working with branches, i.e. derivates of a project



Large file storage

To version (reasonably) large files (images, data samples) → Git with LFS extension.

⚠ Warning

Make sure that the remote host is compatible with LFS (GitHub is compatible)

- Type `git lfs install` to activate the extension
- Create a `data.csv` file and add `Year,Size,Species`
- Type `git lfs track "*.csv"`

A `.gitattributes` file has appeared, which list all the file extensions managed by Git LFS.

Large file storage

- Type `git add .gitattributes data.csv`
- Type `git commit -m "Using LFS"`
- Type `git push`
- On GitHub, click on your file `data.csv` file.

Creating aliases

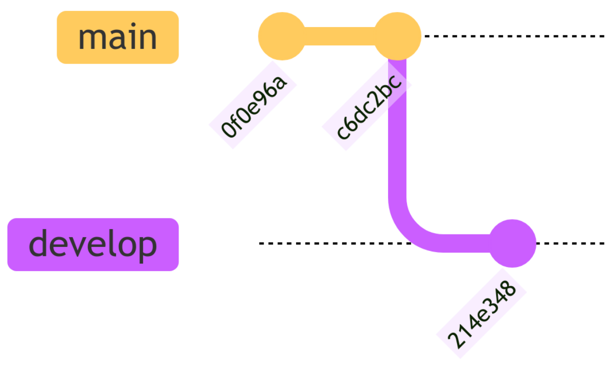
To create Git aliases (i.e. shortcuts):

- Type `git config --global alias.tree log --all --decorate --oneline --graph`
- Type `git config --global alias.br branch -vv`
- Type `git config --global alias.re remove -vv`

You can now call the `git tree`, `git br` and `git re` commands.

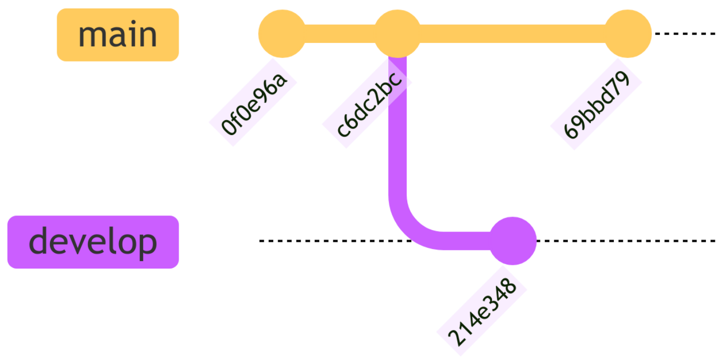
Creating branches

- Type `git checkout -b develop`
- Type `git status`, `git br` and `git tree`
- Open the `README.md` file, add some text and save.
- Type `git add README.md`
- Type `git commit -m "3rd commit"`
- Type `git br` and `git tree`



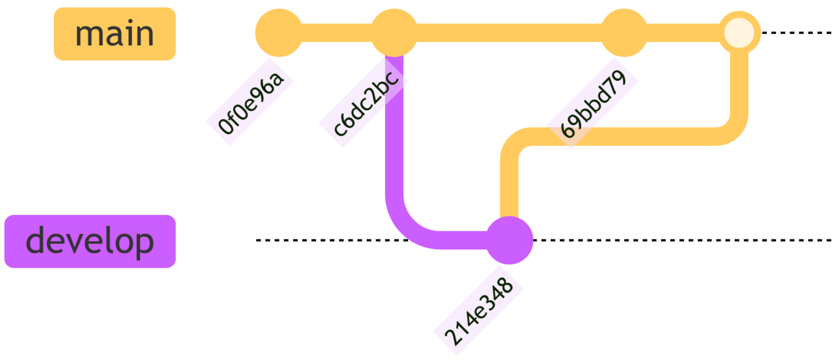
Switching branch

- Type `git checkout main` (or `git checkout master`)
- Type `git br`
- Open the `LICENCE` file and add some text in it
- Type `git add LICENCE`
- Type `git commit -m "Third commit"`
- Type `git tree`



Merging branches

- On the `main` branch, type `git merge develop -m "merge-develop"`
- Type `git log` and `git tree`

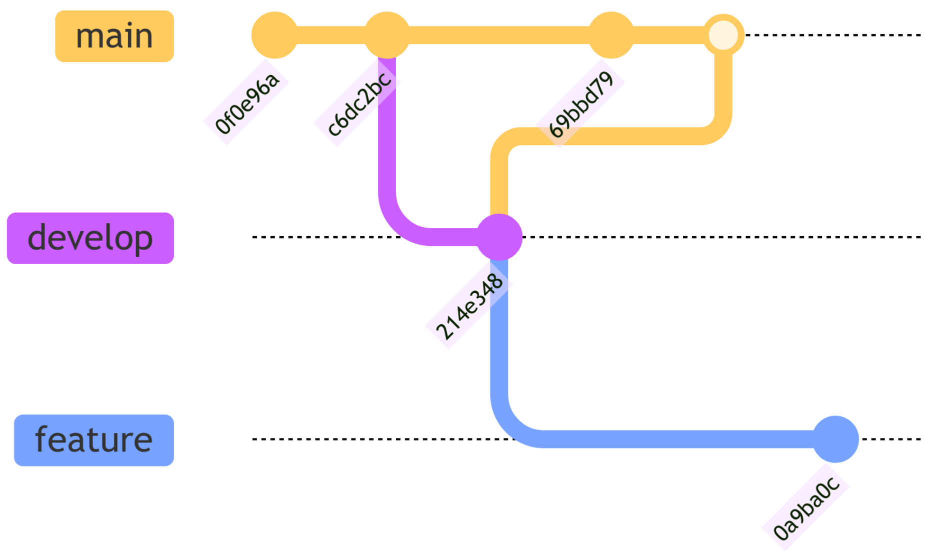


The `merge` command puts the commits from the argument branch (here `develop`) and puts them into the current branch (here `main`).

> **Note**
>
> During the merging process, another commit is created

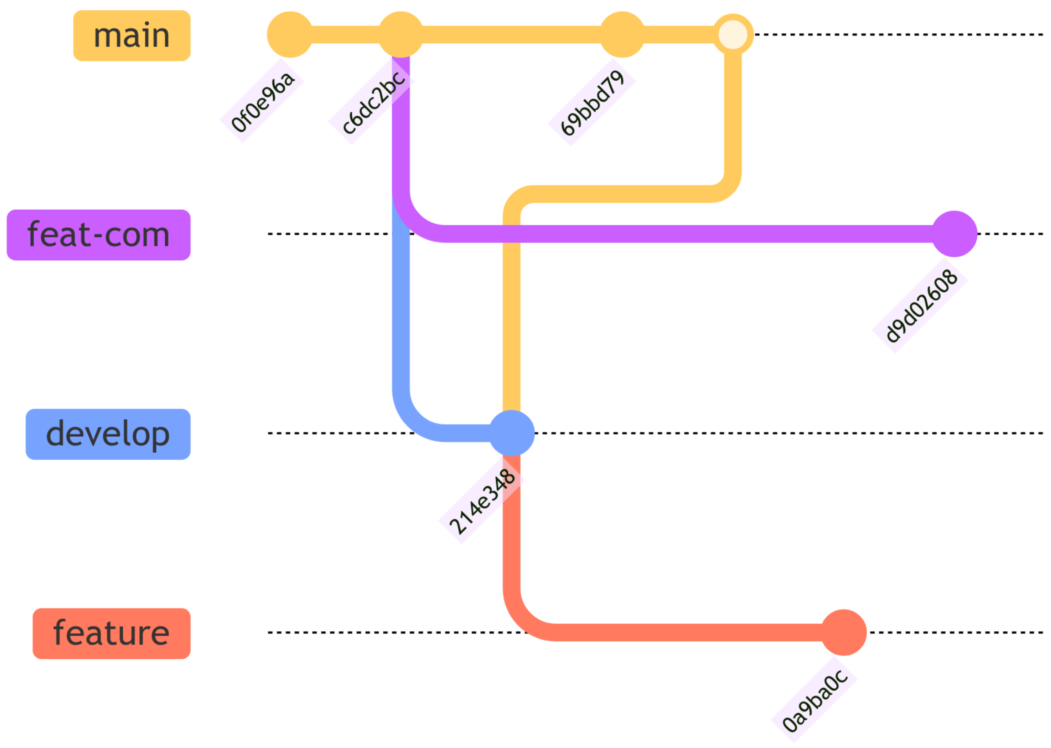
Creating branch from another branch

- Type `git checkout -b feature develop`
- Create a `script.R` file
- Type `git add script.R`
- Type `git commit -m "Fourth commit"`



Creating branch from a commit

- Type `git checkout -b feat-com 1831e4e` replacing `1831e4e` by an actual commit ID.
- Create a `script.py` file
- Type `git add script.py` and `git commit -m "Sixth commit"`



Differences between branches

- Type `git diff develop main`

You will see the text that has been added to the LICENCE file (69bbd79 commit)

> ⚠ Warning
>
> Order matters: it shows what has been added to `main` branch compared to the `develop` branch

Develop
git diff
Main

Deleting a branch

- Type `git checkout main`
- Type `git branch -d develop`
- Type `git br`
- Type `git branch -d feat-com`

An error occurs! The suppression of `feat-com` implies the loss of the `d9d02608` commit. To force the suppression, use `-D` instead of `-d`.

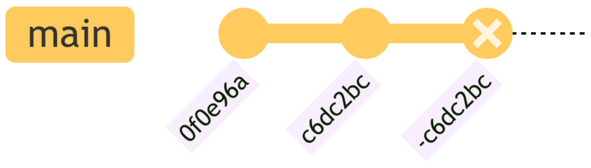
- Type `git branch -D feat-com`

Note

The suppression of `develop` was ok because the content of commit `3rd` is included in the merge.

Reverting a commit

- Type `git revert c6dc2bc` (replace `c6dc2bc` by your commit id)



Remainder

Basic commands

- `git init`: initialise a git project (create .git folder)
- `git add [files]`: add files to list of tracked files
- `git commit -m "message"`: validate locally a version of the project
- `git status`: see the unvalidated and untracked changes
- `git checkout [commit]`: load the project version corresponding to the index
- `git pull`: import the changes from remote project to local
- `git push`: export the changes from local project to the remote server

Git configuration (mandatory)

- Configure your identity: `git config --global user.name "Firstname Lastname"`
- Configure your e-mail: `git config --global user.email "myadresse@ird.fr"`

Branch handling

- `git branch [branch_name]`: create a new branch (but you remain on the previous branch)
- `git branch -b [branch_name]`: create a new branch and move to this newly created branch

- `git checkout [branch_name]`: move to the corresponding branch
- `git merge [branch_name1] [branch_name2]`: merge two different branch, you may need to resolve version conflict.
- `git branch -d [branch_name1]`: delete a branch (safe mode)
- `git branch -D [branch_name1]`: delete a branch (unsafe mode)

Linking with remote

- `git clone [URL]`: Import an existing project from remote server.
- `git remote add origin [URL]`: link directly the local repository with a remote

Authentication of your computer and the remote server

- SSH key: easy way on Linux distributions
  - Tuto here: https://jdblischak.github.io/2014-09-18-chicago/novice/git/05-sshkeys.html
- Authentication Token
  - Tuto here: https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/creating-a-personal-access-token

Good practices

- Pull before any work on the project

- Commit as frequently as possible
- Write explicit commit message
- Push regularly

IDE (graphical user interface) with Git

- R
    - RStudio
    - Visual Studio Code
- Python
    - Spyder
    - Visual Studio Code
    - Pycharm (all JetBrain softwares)

Sources

- Plateau bioinformatique, Montpellier: Formation Git(Lab) (05/04/2018)
- UMR AMAP (Atelier MAIA P3M), Montpellier: Introduction à GIT (04/04/2019)