

Practical session: Version control, Git, GitHub and GitFlow

Criscely Luján^{1,2}
criscely.lujan@ird.fr

Nicolas Barrier²
nicolas.barrier@ird.fr

¹Université Paris-Sud, UMR MARBEC

²IRD, UMR MARBEC

April 11, 2019





Table of Contents

1 Git basis (beginners)

2 Git Flow (advanced)

GitHub account

Tips about the name account:

- Use your actual name!
- Shorter is better than longer!
- Be as unique as possible!
- Re-use your name from other context, e.g. , , !

Git is already installed?

To check that go to shell (terminal / command line / console) and enter **which git** to request the path to your Git executable:

```
which git
```

Then enter **git --version** to see its version:

```
git --version
```

If git is not installed YET: See [Install git](#) to follow the correct steps to install git according your system operative! :)

Introduce yourself to Git

Let **git** to know about you, following this simple configuration steps!

```
# Example
git config --global user.name "Criscely Lujan"
git config --global user.email "criscelylujan@gmail.com"
git config --global core.editor vim
git config --global --list
```

- **user.name** can be your username. Your commits will be labelled with this name, so this should be informative!
- **user.email** must be the email that you use to sign up for GitHub.
- **core.editor** There are diverse options of [Git editor](#).

How authenticating yourself with GitHub

There are two options of protocols for secure communication working over a computer network!

1. **Hypertext Transfer Protocol Secure (HTTPS):**

If you plan to work using HTTPS protocol, you can follow [*Cache credential for HTTPS*](#) for more information.

2. **Secure Shell (SSH):**

If you plan to work using SSH protocol, you can follow [*Set up keys for SSH*](#) for more information.

Exercise 1: Cloning a repository

- Go to the repository of the Git training:
<https://github.com/umr-marbec/git-training>.
- Explore the repository: number of commits, contributors, stars,
- Have a look of the README file.

Exercise 1: Cloning a repository using terminal

- Create a new directory, open it in terminal and perform the following code:

```
git init # to create a new git repository
```

- Clone the repository running the following command plus the path of the repository to be cloned:

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

This path is copied from the repository that we will be cloned. You now have the class content on your computer.

Exercise 2: Creation of a repository

- Go to GitHub and login. Click in the green box called **New repository**.
- If you are on your own profile page, go to the section **Repositories**, then click the green box called **New**.
- Assign a name for the **repository** and include a **description** (this is optional but is recommended!).
- Public or private.
- Initialize the repository without **without the README**.

Exercise 2: Creation of a repository

Create a new repository

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

Owner

Repository name *



CriscelyLP ▾

/

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.

Add .gitignore: **None** ▾

Add a license: **None** ▾



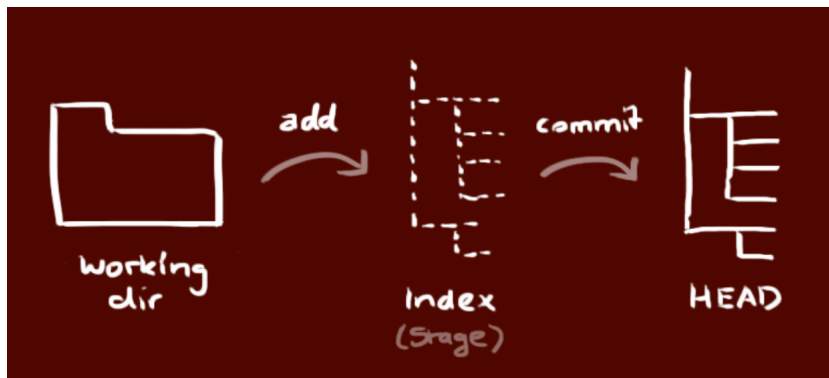
Create repository

Exercise 2: Creation of a repository

- Go to your GitHub profile and see the section called Repositories!.
- Clone the repository already created (following steps in exercise 1) in your computer!

Workflow

Your local repository consists of three trees maintained by git.



Exercise 3: Make changes in the cloned repository

- After clone the repository on your computer, you are able to make changes using **add**, **commit**, and **push**: - Create a README.md file (similar to the one in the git-training repo.)
- Now add the file to the distant repository as follows:

```
git add README.md # adding changes for specific file
git commit -m "Commit message" # changes committed to the HEAD
git push origin master # changes to the remote repository
```

- Check the changes in your remote repository.

Note

To stage all the files of the current directory:

```
git add .
```

Exercise 4: Create branches

- Create a branch from the master one:

```
git checkout -b newbranch
```

- Put the branch on GitHub

```
git push origin newbranch
```

- Check on the GitHub web page if your branch appears. To check the branches:

```
git branch
```

- Edit the README.md file, stage, commit and push changes following the steps of the previous slide. To push on the new branch:

```
git push origin newbranch
```

- Check that the file has been updated in the branch.

Exercise 5: Merge branches

Now, the modification provided in the `newbranch` will be moved back to the main branch.

- Go back to the main branch

```
git checkout master  
git branch # checks that we are on the master branch
```

- Merge the newbranch into master

```
git merge newbranch
```

- Push the updated master branch

```
git push origin master
```

- Remove the newbranch, which is no more useful

```
git branch -d newbranch # remove local branch  
git push origin --delete newbranch # remove from distant  
repo
```

Managing conflicts

- Clone your repository in another location
- From this new repository, edit the README file and push the change
- Go back to your first repo., edit the README file and push the change. **You should see an error!**
- Do an update from the distant branch

```
git pull  
git status
```

- Resolve conflict, add and commit changes.
- Push the resolved master branch

```
git pull origin master
```


See the life of your branches

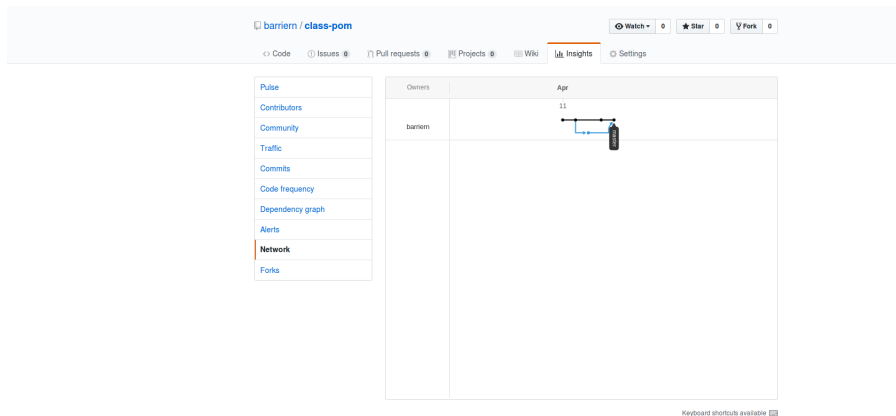


Table of Contents

1 Git basis (beginners)

2 Git Flow (advanced)

Git Flow (advanced users)

Install the GitFlow extension:

```
sudo apt-get install git-flow
```

Now, from your directory, initialize the GitFlow workflow:

```
git flow init # create the develop branch  
git push origin develop # push the develop branch to the  
repository
```

Creating new features

```
# recovers the latest develop branch
git pull origin develop

# create feature branch from develop and switch to it
git flow feature start my-feature

# put the feature branch to the repo (not necessary)
git flow feature publish my-feature

# edit/add files: new functionalities
# commit changes to the my-feature branch
git commit

# merge my-feature branch with develop and remove it
# switch back to develop
git flow feature finish

# push updated develop branch
git push origin develop
# if conflict (someone has already pushed in develop)
git pull origin develop
git push origin develop
```

Creating a new release

```
# recovers the latest develop branch
git pull origin develop

# create a new release branch
git flow release start my-release

# put the release branch to the repo (not necessary)
git flow release publish my-release

# edit/add files: bug fixes + documentation

# commit changes to the my-feature branch
git commit

# finish the release
git flow release finish

# push updated develop/master and tags
git push origin --tags
git push origin develop
git push origin master
```

Creating hotfixes

```
# recovers the latest develop/master branches
git pull origin develop

# create a new hotfix branch from master
# name should be a version name (1.3.2 for instance)
git flow hotfix start hotfix-tag

# edit/add files: bug corrections

# commit changes to the hotfix-tag branch
git commit

# finish the release
# merge hotfix branch to master/develop
git flow hotfix finish

# push updated develop/master and tags
git push origin --tags
git push origin develop
git push origin master
```