Introduction to Git

DataTrek 2021

Présentation

Noel Rignon

B.Ing Genie Logiciel

CEO at FJNR
CTO at ExoFreeMotion

FJNR

Exo
Free
Motion

Github: RignonNoel

LinkedIn: rignonnoel

Summary of the course

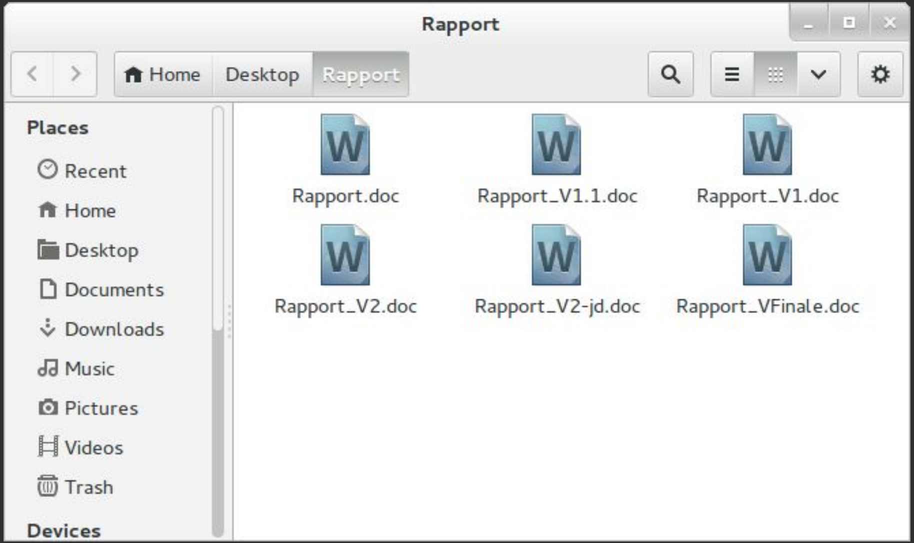
Introduction to Git

1. Why Git ?
2. What is Git ?
3. Terminology
4. How to use Git ?
5. Exercises
6. Ressources

Why Git ?

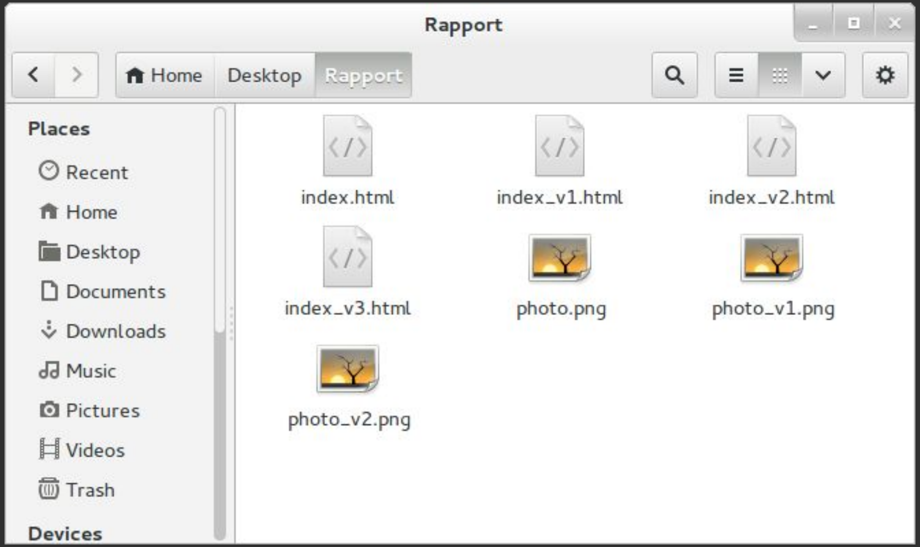
Let's remember some frustrating things together

How to keep track of versions ?



Source: https://perso.liris.cnrs.fr/pierre-antoine.champin/enseignement/intro-git/

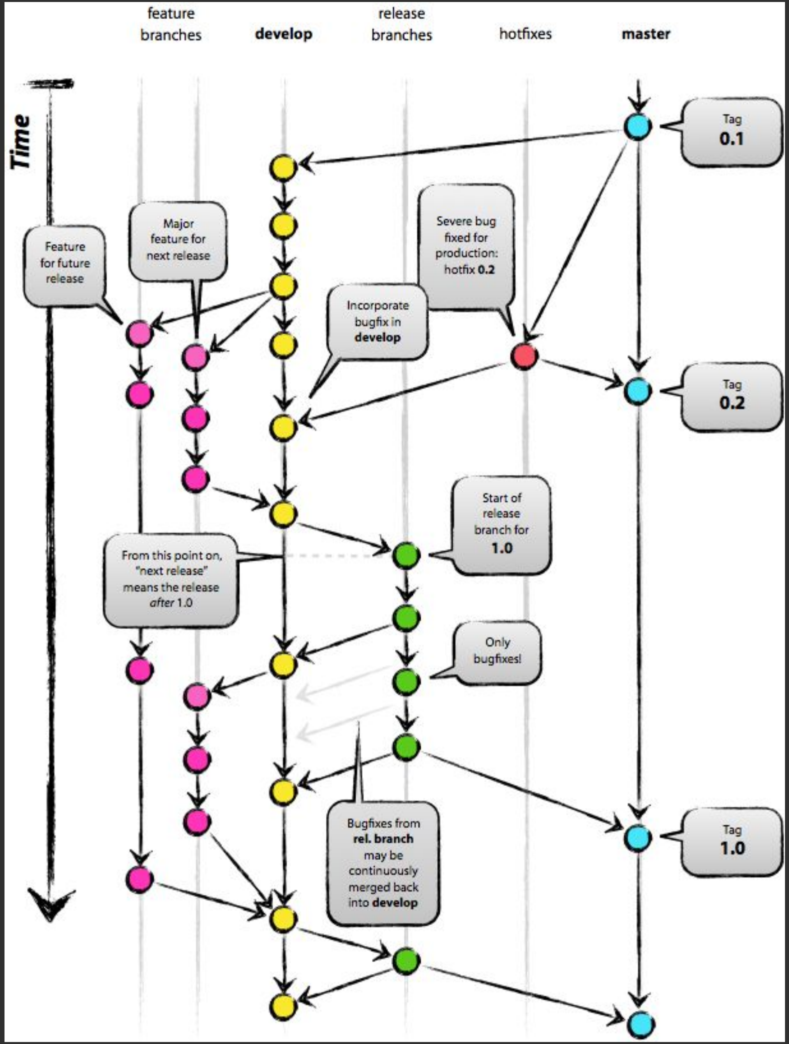
How to keep track of the relations ?



Source: https://perso.liris.cnrs.fr/pierre-antoine.champin/enseignement/intro-git/

A little example of one Git Workflow

- people can work in parallel
- changes can be incorporated on differents version
- changes can be discussed and review before approval
- everything is clear for each participant


feature branches
develop
release branches
hotfixes
master
Time
Tag 0.1
Feature for future release
Major feature for next release
Severe bug fixed for production: hotfix 0.2
Incorporate bugfix in develop
Tag 0.2
Start of release branch for 1.0
From this point on, "next release" means the release after 1.0
Only bugfixes!
Bugfixes from rel. branch may be continuously merged back into develop
Tag 1.0

What is Git ?

A little bit of common knowledge
to shine in society

Introduction

- Created in 2005 by Linus Torvalds (Developer of the Linux kernel)

- Distributed version-control

Goal

- Speed
- Data integrity
- Support for distributed non-linear workflows

License

GNU General Public License V2

- Freedom 0: The freedom to run the program for any purpose.
- Freedom 1: The freedom to study how the program works, and change it to make it do what you wish.
- Freedom 2: The freedom to redistribute and make copies so you can help your neighbour.
- Freedom 3: The freedom to improve the program, and release your improvements (and modified versions in general) to the public, so that the whole community benefits.

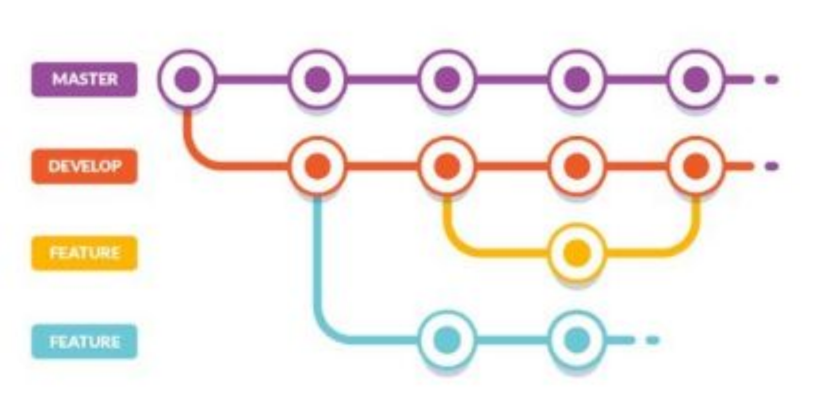
Terminology

Commit

A commit is like a “revision” of your document, it contain all the changes you made (addition, deletion)

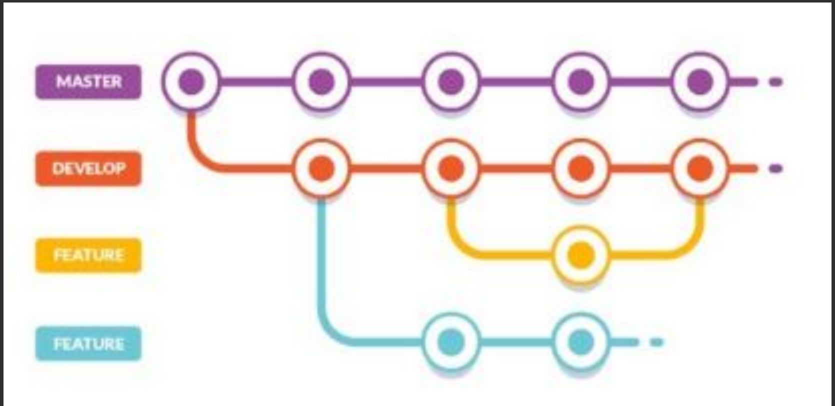
Each commit contain:

- A date
- An author
- A description
- The list of changes
- Some links to others commits



Branch

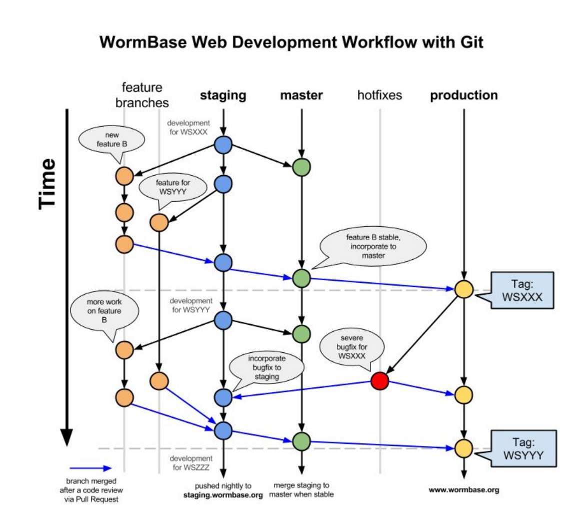
A branch is a structure created from multiple commits depending on each others.



Repository

A repository is kind of a “project”, you will work and add all your changes in it.

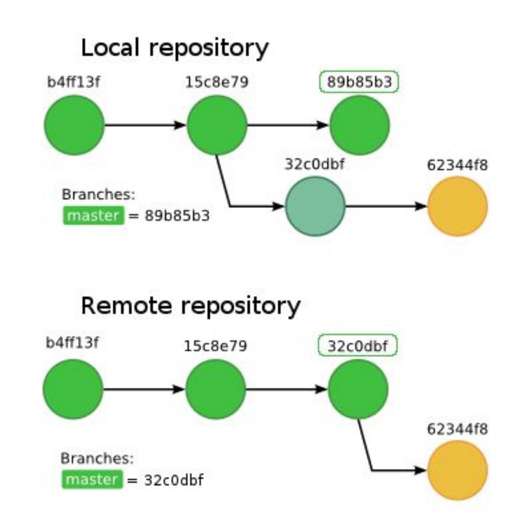
It will contain all your branches and commits.


WormBase Web Development Workflow with Git
feature branches
staging
master
hotfixes
production
Time
development for WSXXX
new feature B
feature for WSYYY
feature B stable, incorporate to master
Tag: WSXXX
more work on feature B
development for WSYYY
severe bugfix for WSXXX
incorporate bugfix to staging
Tag: WSYYY
development for WSZZZ
branch merged after a code review via Pull Request
pushed nightly to staging.wormbase.org
merge staging to master when stable
www.wormbase.org

Remote

GitHub is an easy way to store your remote repository on the cloud

A “remote” is a distant repository, it allow you to sync your work with other people via the network and help manage potential conflict.

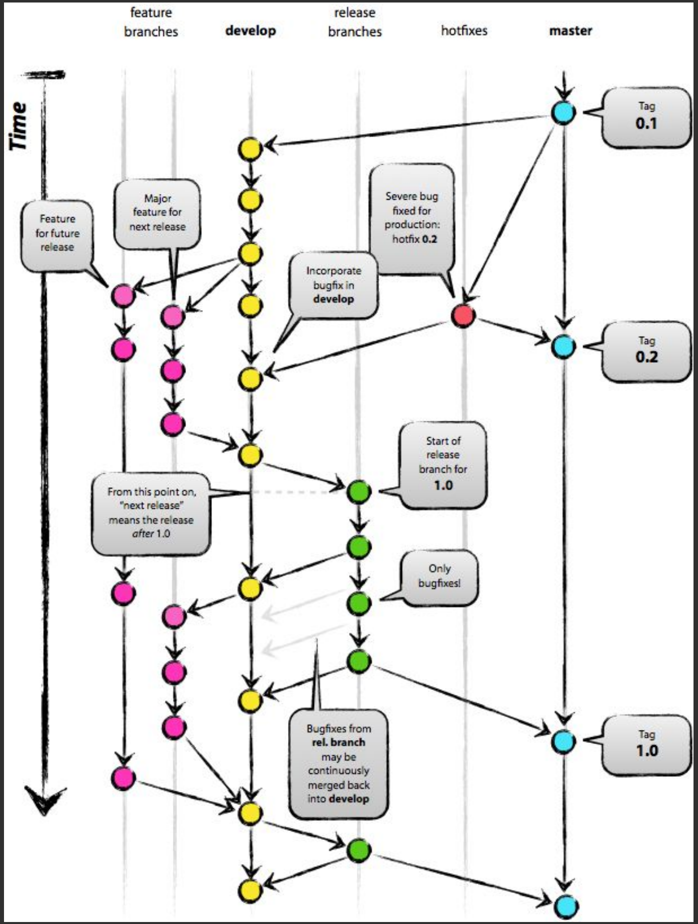


How to use Git ?

Finally! The hardest part..

Different workflow

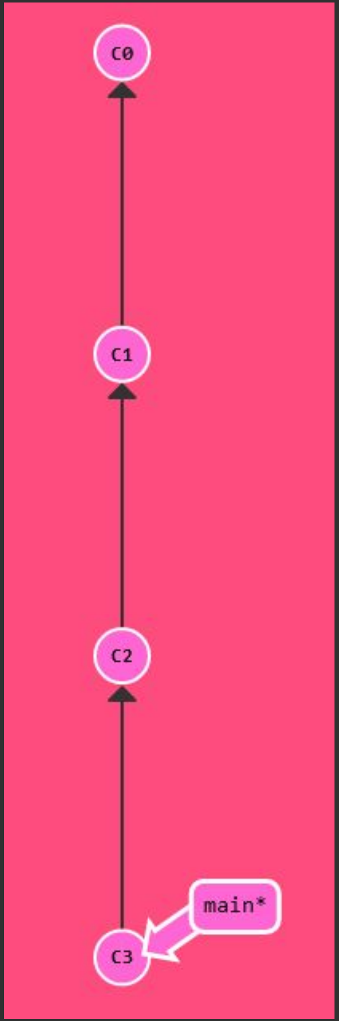
You should define with your team what is the best way to work together


feature branches
develop
release branches
hotfixes
master
Time
Tag 0.1
Feature for future release
Major feature for next release
Severe bug fixed for production: hotfix 0.2
Incorporate bugfix in develop
Tag 0.2
Start of release branch for 1.0
From this point on, "next release" means the release after 1.0
Only bugfixes!
Bugfixes from rel. branch may be continuously merged back into develop
Tag 1.0

Beginner workflow

Avoid conflict and parallelization the time to ramp-up

Do not forget to ramp-up ;)



Cloning a repository

git clone [url]

There are two way to begin a repository:

1. git init: to initialize a new one from scratch

2. git clone [url]: to clone an existing repository created on Github for example

```
> git clone https://github.com/RignonNoel/partage
```

By default, your cloned remote will be named “origin”

Add a remote

```
git remote add [name] [url]
git remote -v
```

If you need to work with others distant repository than the one you cloned, you can configure some remote in local.

For example if you cloned your personal repository, you can add a remote to the official repository.

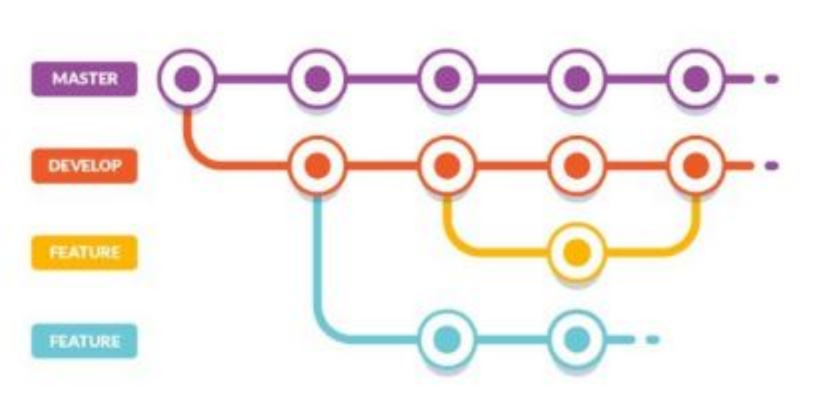
```
> git clone https://github.com/RignonNoel/partage

> git remote add upstream
https://github.com/randonneesdatatrek/partage
```

Change branch

git checkout [remote] [branch]

You can change the current branch you work on by doing a “checkout”

```
> git checkout origin develop
```

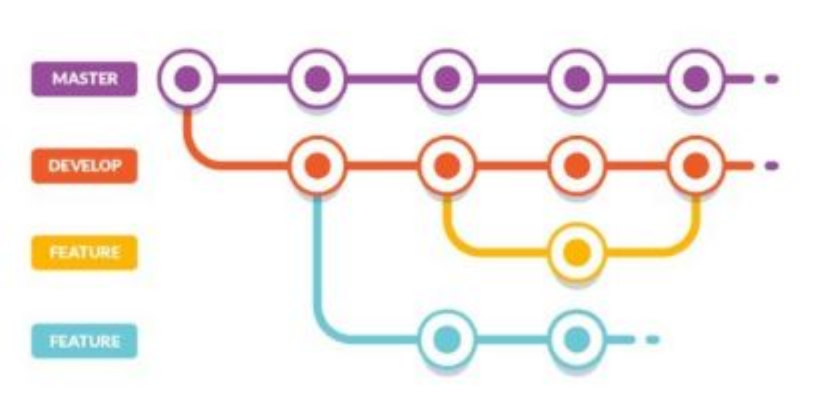


Create a new branch

git checkout -b [remote] [branch]

You can create a new branch from your current position in order to begin a parallelized work

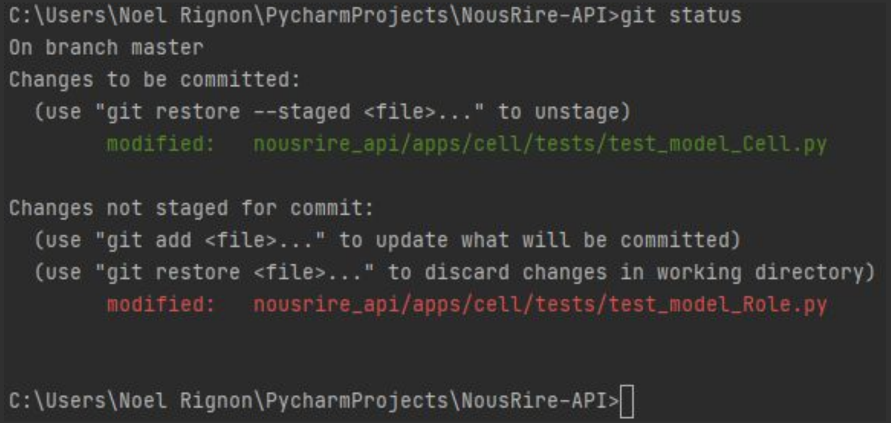
```
> git checkout -b my-new-feature
```



Add a new commit

git status
git add [file]
git commit -m “[message]”

In order to create a new commit you need to “stage” your change:



```
C:\Users\Noel Rignon\PycharmProjects\NousRire-API>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   nousrire_api/apps/cell/tests/test_model_Cell.py

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   nousrire_api/apps/cell/tests/test_model_Role.py


C:\Users\Noel Rignon\PycharmProjects\NousRire-API>
```

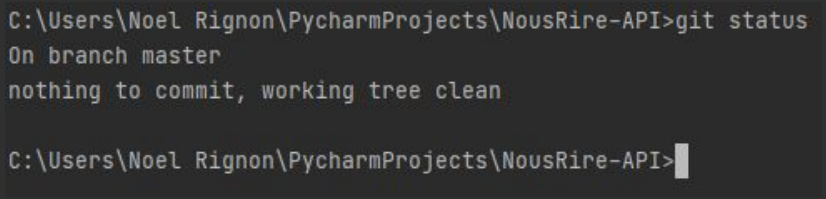
```
> git status
> git add nousrire_api/apps/cell/tests/test_model_Role.py
> git status
> git commit -m “add tests on cell and role models”
```

Push your change

git push [remote] [branch]

When your changes are committed and ready to share with your team* you can push them to your remote:

```
> git push origin my-new-feature
```



```
C:\Users\Noel Rignon\PycharmProjects\NousRire-API>git status
On branch master
nothing to commit, working tree clean

C:\Users\Noel Rignon\PycharmProjects\NousRire-API>
```

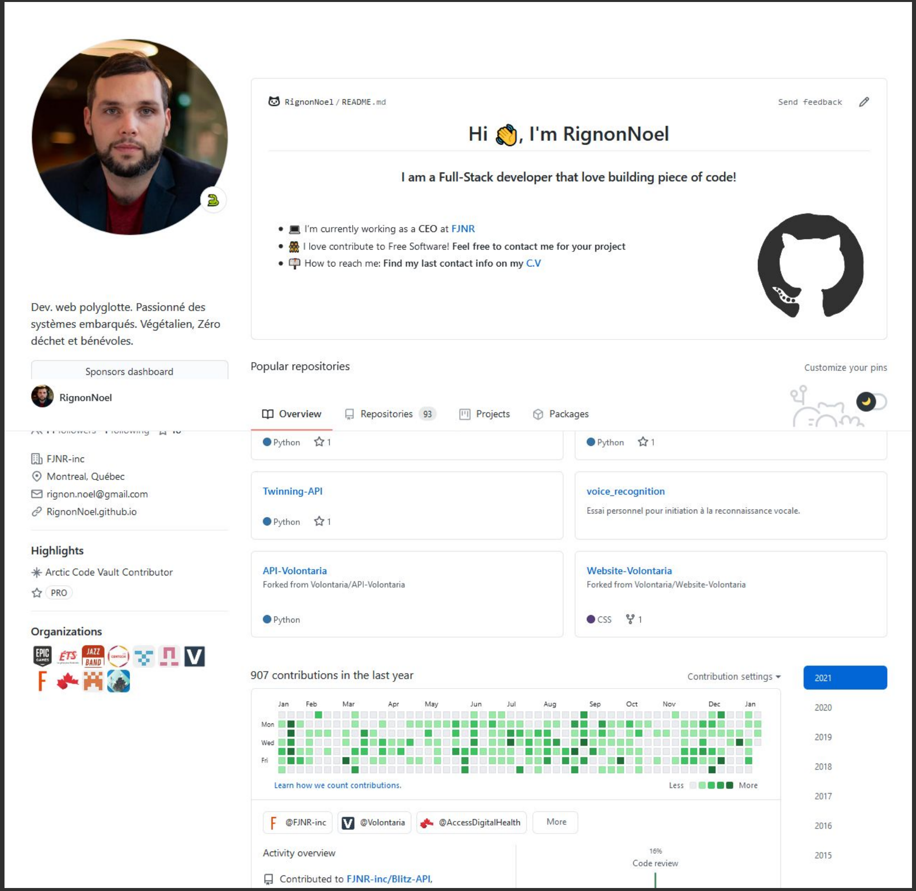
* *Or ready to save on Github to have a secure copy and not lost all you work with you laptop in your next housebreaking*

Exercises

And some bonus to
be a Git Warrior

Create your Github account

https://github.com/



RignonNoel / README.md

Send feedback

Hi 👋, I'm RignonNoel

I am a Full-Stack developer that love building piece of code!

- 💻 I'm currently working as a CEO at FJNR
- 👯 I love contribute to Free Software! **Feel free to contact me for your project**
- 📫 How to reach me: **Find my last contact info on my** C.V

Dev. web polyglotte. Passionné des systèmes embarqués. Végétalien, Zéro déchet et bénévoles.

Sponsors dashboard

RignonNoel

FJNR-inc
Montreal, Québec
rignon.noel@gmail.com
RignonNoel.github.io

Highlights

✳ Arctic Code Vault Contributor
☆ PRO

Organizations

Popular repositories

Customize your pins

Overview | Repositories 93 | Projects | Packages

Python ☆ 1

Python ☆ 1

Twinning-API
Python ☆ 1

voice_recognition
Essai personnel pour initiation à la reconnaissance vocale.

API-Volontaria
Forked from Volontaria/API-Volontaria
Python

Website-Volontaria
Forked from Volontaria/Website-Volontaria
CSS 1

907 contributions in the last year

Contribution settings ▾

Learn how we count contributions.

@FJNR-inc | @Volontaria | @AccessDigitalHealth | More

Activity overview

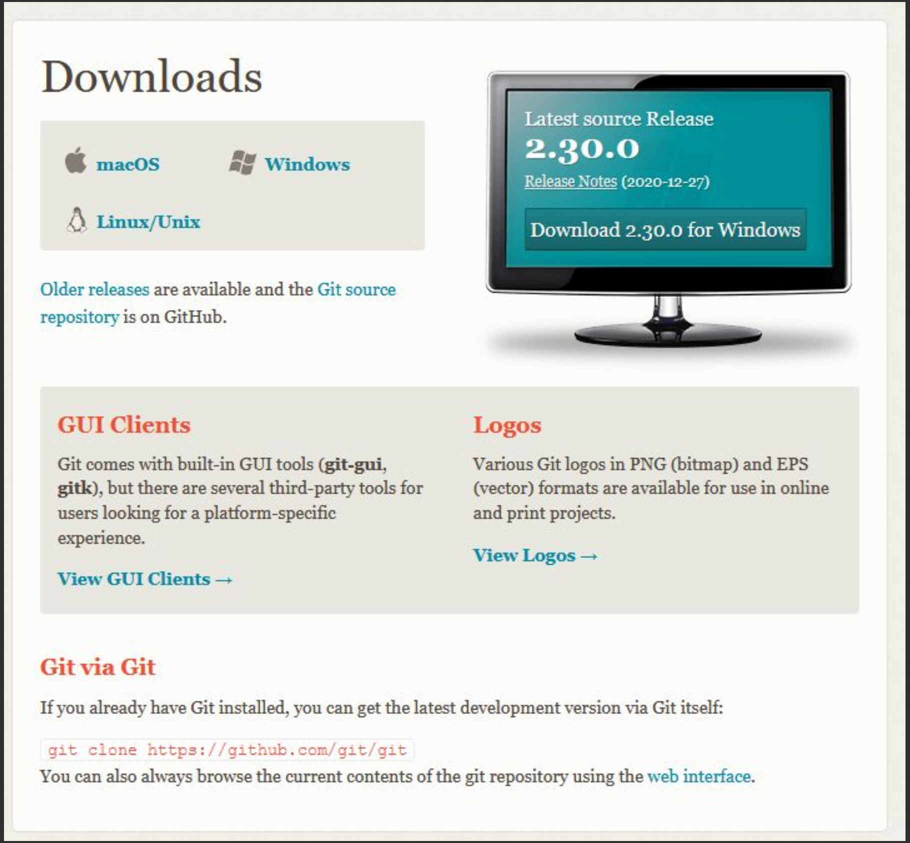
Contributed to FJNR-inc/Blitz-API,

16%
Code review

Install Git on your computer

https://git-scm.com/downloads

https://git-scm.com/downloads/guis



Downloads

Older releases are available and the Git source repository is on GitHub.

Latest source Release
2.30.0
Release Notes (2020-12-27)

GUI Clients

Git comes with built-in GUI tools (**git-gui**, **gitk**), but there are several third-party tools for users looking for a platform-specific experience.

View GUI Clients →

Logos

Various Git logos in PNG (bitmap) and EPS (vector) formats are available for use in online and print projects.

View Logos →

Git via Git

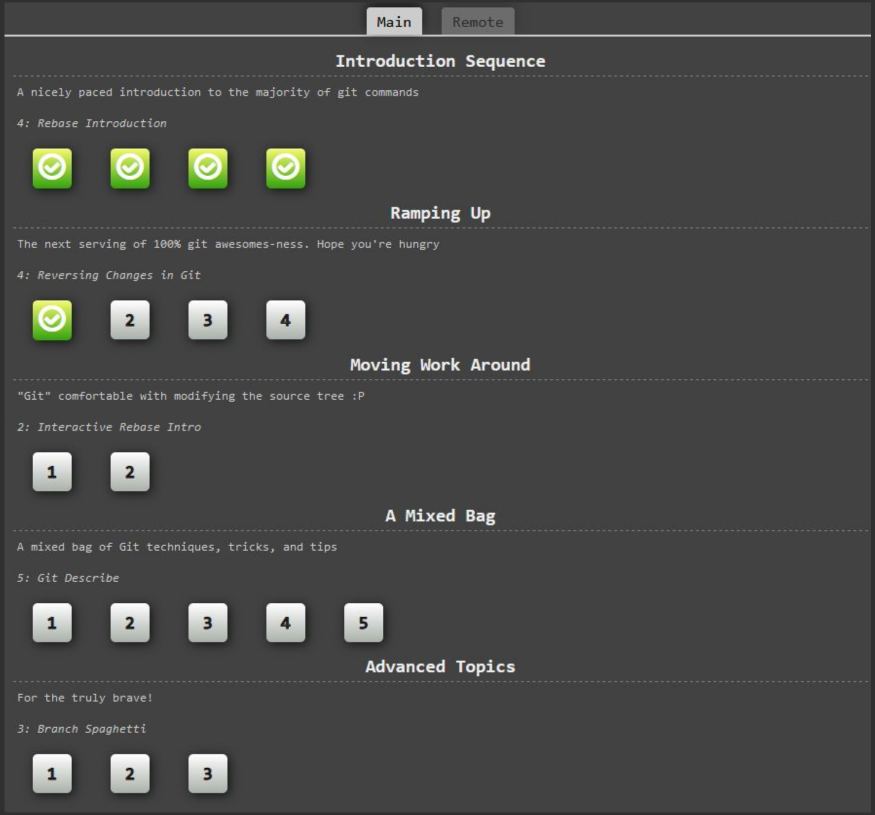
If you already have Git installed, you can get the latest development version via Git itself:

```
git clone https://github.com/git/git
```

You can also always browse the current contents of the git repository using the web interface.

Introduction sequence

https://learngitbranching.js.org/



Introduction Sequence

A nicely paced introduction to the majority of git commands

4: Rebase Introduction

Ramping Up

The next serving of 100% git awesomes-ness. Hope you're hungry

4: Reversing Changes in Git

Moving Work Around

"Git" comfortable with modifying the source tree :P

2: Interactive Rebase Intro

A Mixed Bag

A mixed bag of Git techniques, tricks, and tips

5: Git Describe

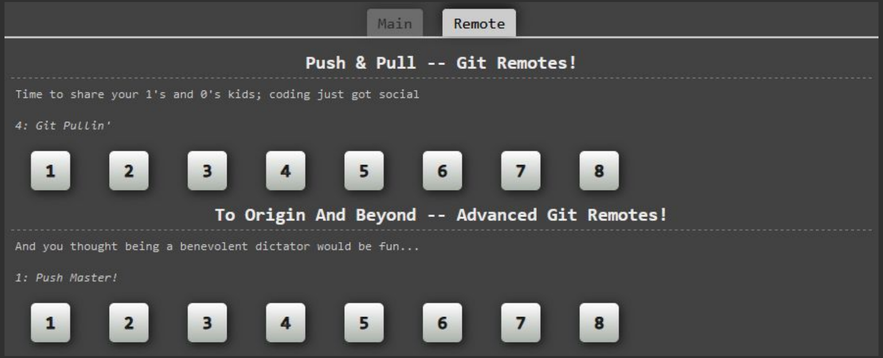
Advanced Topics

For the truly brave!

3: Branch Spaghetti

Learn Git branching

https://learngitbranching.js.org/



Push & Pull -- Git Remotes!

Time to share your 1's and 0's kids; coding just got social

4: Git Pullin'

To Origin And Beyond -- Advanced Git Remotes!

And you thought being a benevolent dictator would be fun...

1: Push Master!

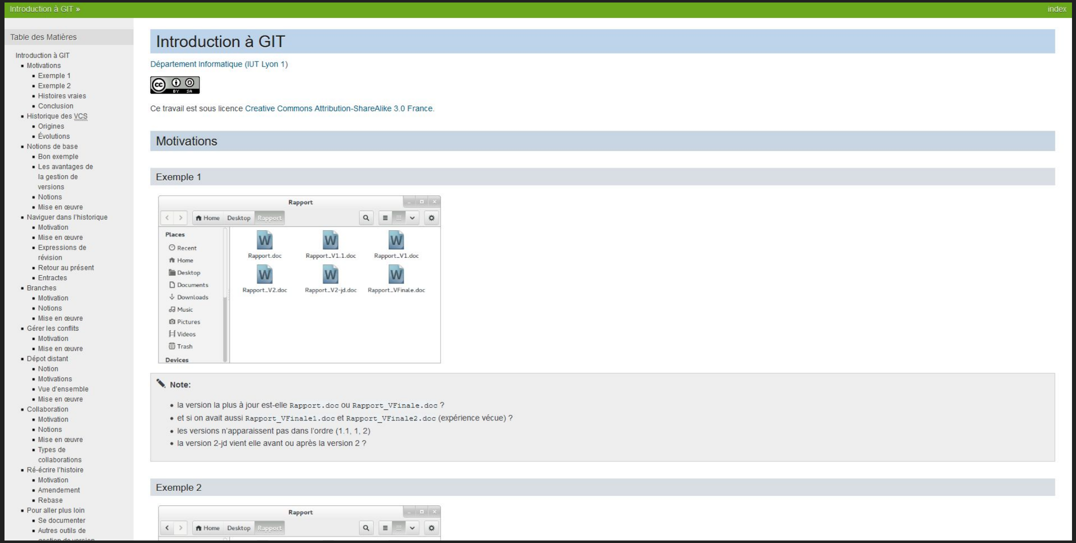
Ressources

Some good links
everybody should know

Ressource: Git introduction

A good Git introduction:

https://perso.liris.cnrs.fr/pierre-antoine.champin/enseignement/intro-git/



Introduction à GIT »

index

Table des Matières

Introduction à GIT

Département Informatique (IUT Lyon 1)

Motivations

Exemple 1

Note:

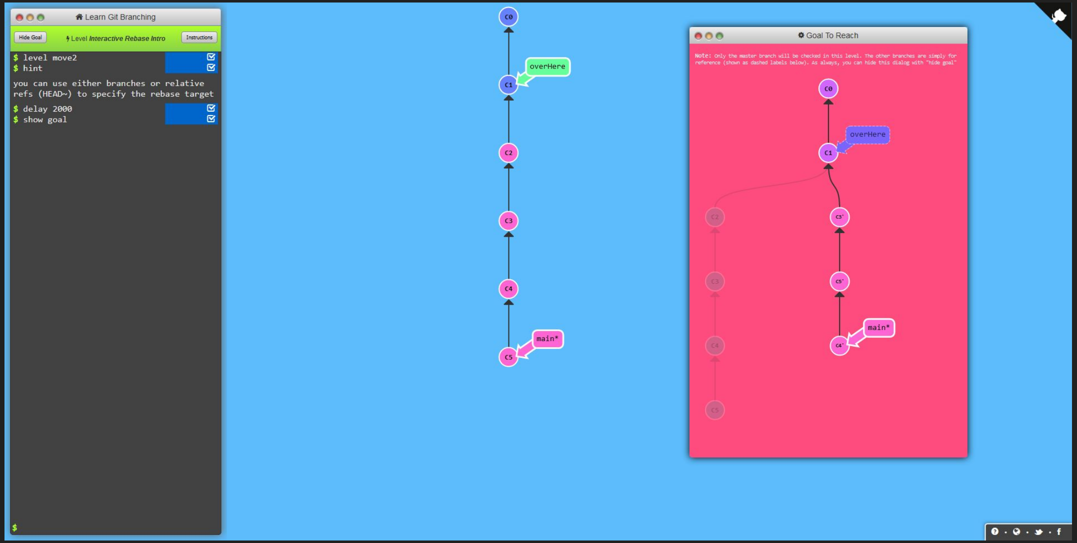
- la version la plus à jour est-elle `Rapport.doc` ou `Rapport_VFinale.doc` ?
- et si on avait aussi `Rapport_VFinale1.doc` et `Rapport_VFinale2.doc` (expérience vécue) ?
- les versions n'apparaissent pas dans l'ordre (1.1, 1, 2)
- la version 2-jd vient elle avant ou après la version 2 ?

Exemple 2

Ressource: Learn Git Branching

A dynamic web-tool to learn git by practice with tutorial:

https://learngitbranching.js.org/



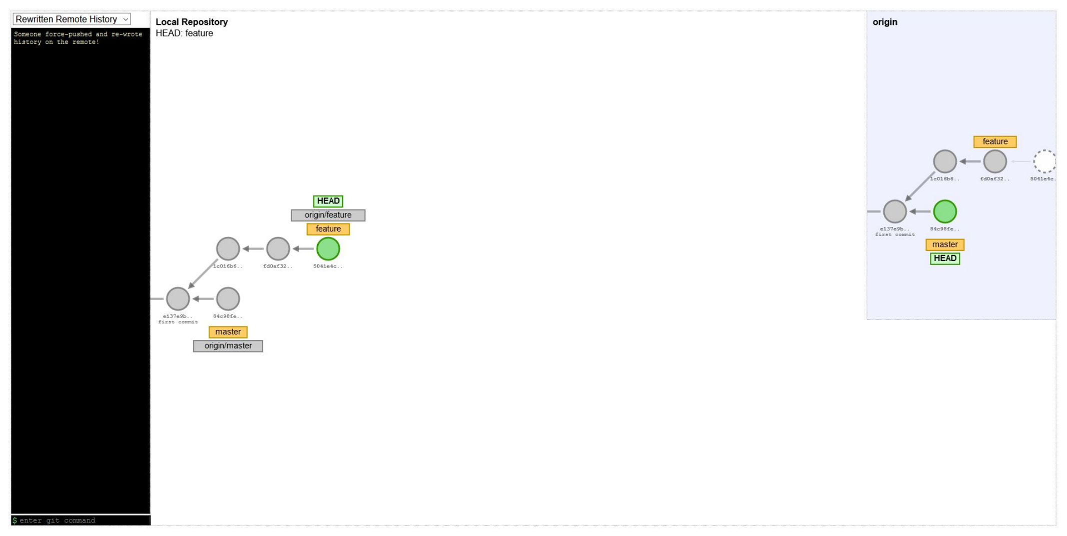
```
$ level move2
$ hint

you can use either branches or relative
refs (HEAD~) to specify the rebase target

$ delay 2000
$ show goal

$
```

Note: Only the master branch will be checked in this level. The other branches are simply for reference (shown as dashed labels below). As always, you can hide this dialog with "hide goal"

Ressource: Git Sandbox

A dynamic web sandbox to try git by practicing (not a tutorial, just a sandbox):

https://git-school.github.io/visualizing-git/#free



Rewritten Remote History

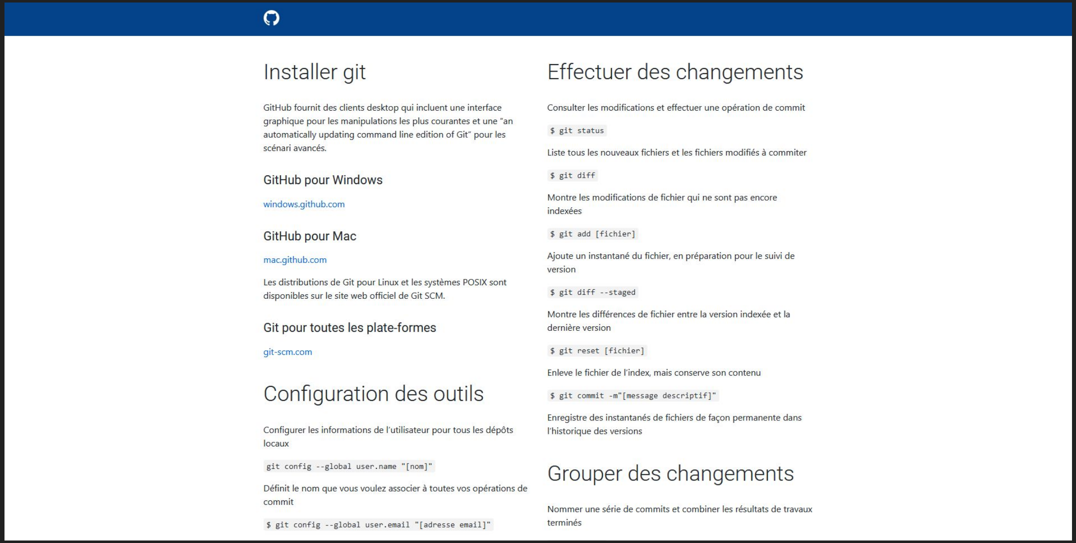
```
Someone force-pushed and re-wrote
history on the remote!
```

```
$ enter git command
```

Ressource: Git Cheat-Sheet

A good multi-lingual git cheat-sheet:

https://training.github.com/



Installer git

GitHub fournit des clients desktop qui incluent une interface graphique pour les manipulations les plus courantes et une "an automatically updating command line edition of Git" pour les scénari avancés.

GitHub pour Windows

windows.github.com

GitHub pour Mac

mac.github.com

Les distributions de Git pour Linux et les systèmes POSIX sont disponibles sur le site web officiel de Git SCM.

Git pour toutes les plate-formes

git-scm.com

Configuration des outils

Configurer les informations de l'utilisateur pour tous les dépôts locaux

```
git config --global user.name "[nom]"
```

Définit le nom que vous voulez associer à toutes vos opérations de commit

```
$ git config --global user.email "[adresse email]"
```

Effectuer des changements

Consulter les modifications et effectuer une opération de commit

```
$ git status
```

Liste tous les nouveaux fichiers et les fichiers modifiés à commiter

```
$ git diff
```

Montre les modifications de fichier qui ne sont pas encore indexées

```
$ git add [fichier]
```

Ajoute un instantané du fichier, en préparation pour le suivi de version

```
$ git diff --staged
```

Montre les différences de fichier entre la version indexée et la dernière version

```
$ git reset [fichier]
```

Enleve le fichier de l'index, mais conserve son contenu

```
$ git commit -m"[message descriptif]"
```

Enregistre des instantanés de fichiers de façon permanente dans l'historique des versions

Grouper des changements

Nommer une série de commits et combiner les résultats de travaux terminés