Git

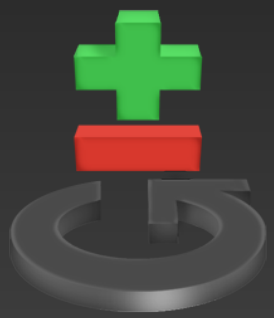
Atelier d'aide à la programmation

Léo Baudouin

baudouin.leo@gmail.com

03-04 juin 2019

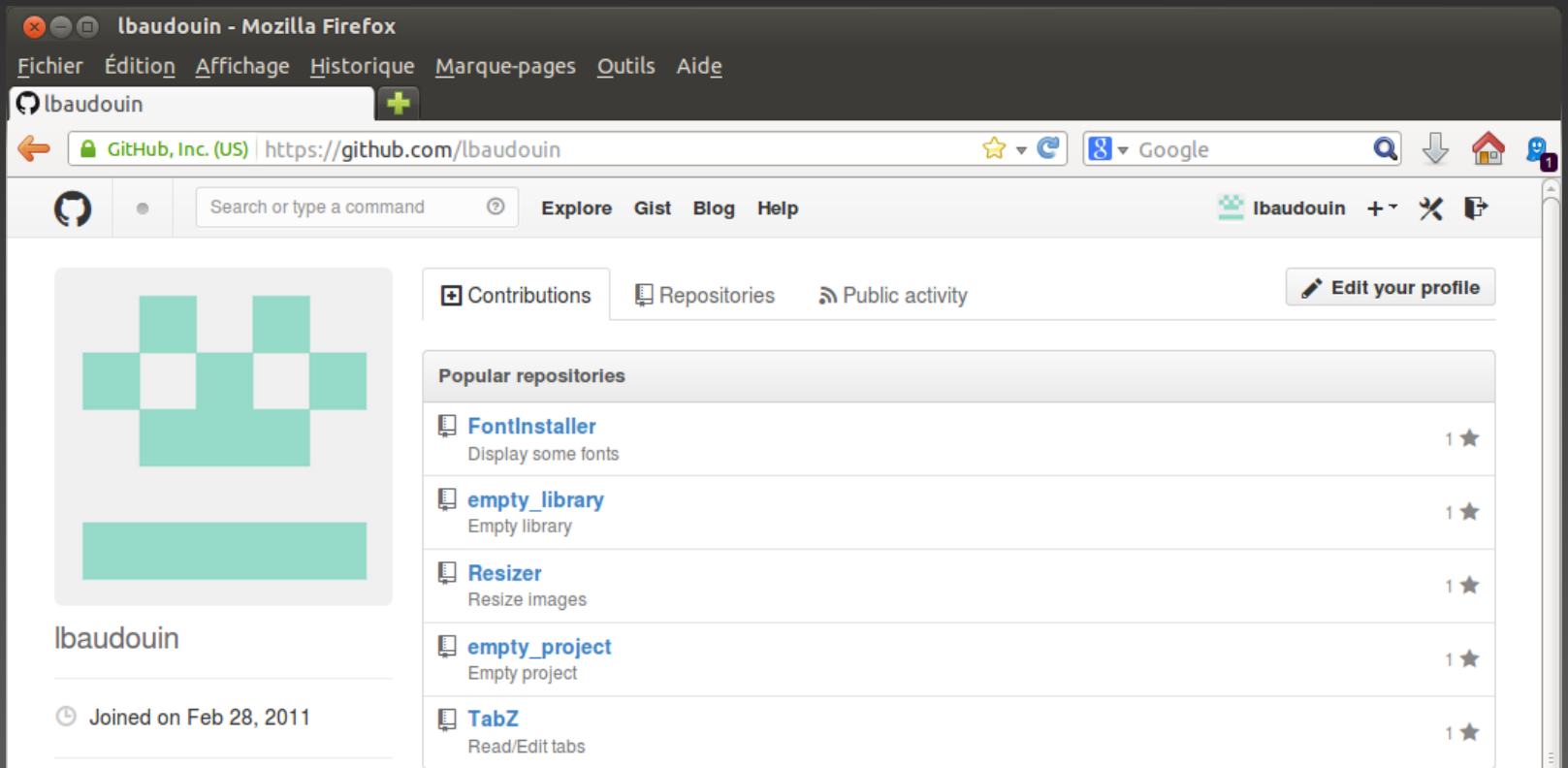
Logiciel de suivi de version



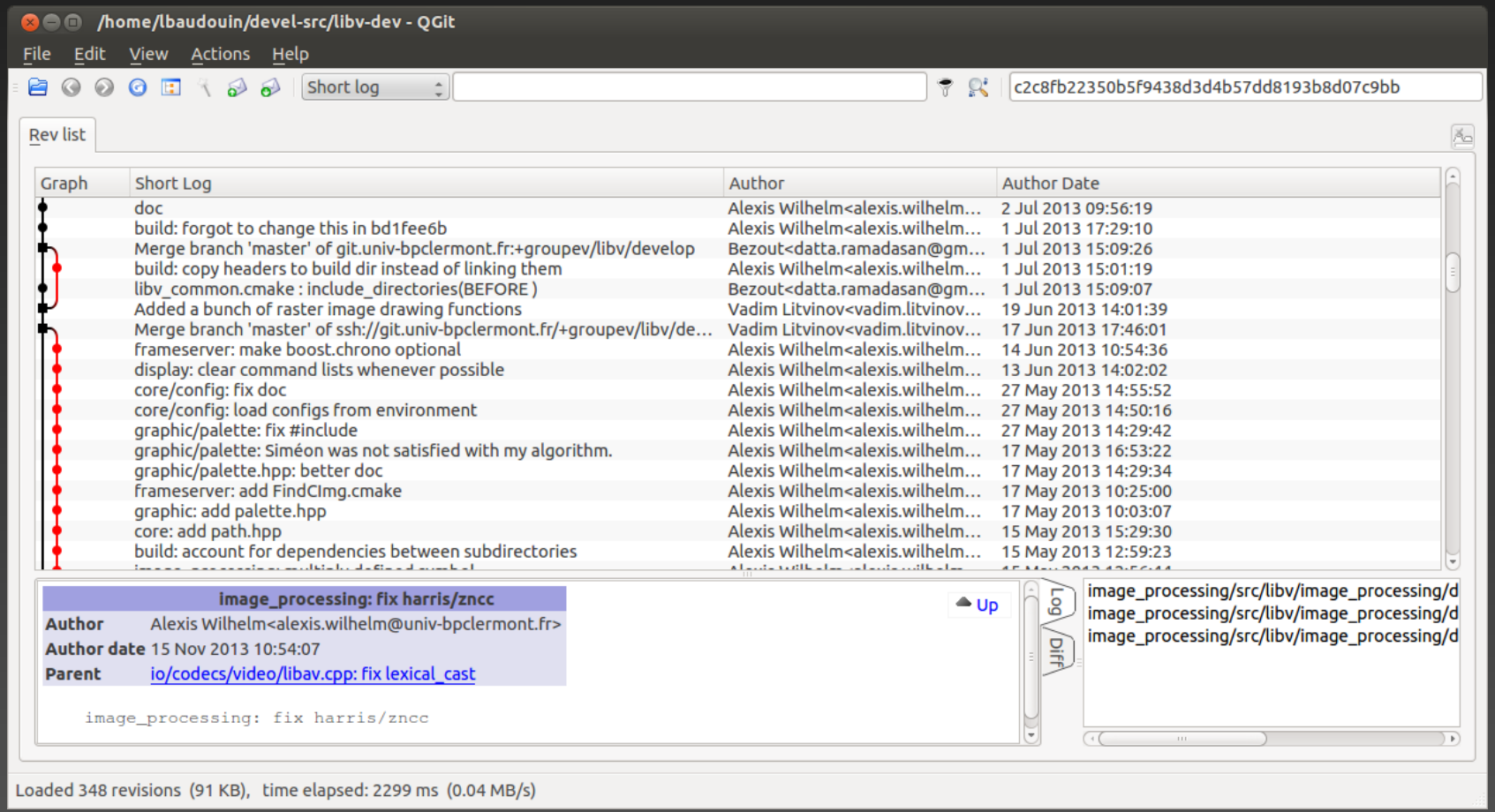
Git *est un logiciel de gestion de versions décentralisé. C'est un logiciel libre créé par* **Linus Torvalds**, *créateur du noyau Linux, et distribué selon les termes de la licence publique générale GNU version 2.*

Logiciel de suivi de version

Git *est un logiciel de gestion de versions décentralisé. C'est un logiciel libre créé par* **Linus Torvalds**, *créateur du noyau Linux, et distribué selon les termes de la licence publique générale GNU version 2.*



Git


/home/lbaudouin/devel-src/libv-dev - QGit
File Edit View Actions Help
Short log
c2c8fb22350b5f9438d3d4b57dd8193b8d07c9bb
Rev list
Graph
Short Log
Author
Author Date
doc Alexis Wilhelm<alexis.wilhelm... 2 Jul 2013 09:56:19
build: forgot to change this in bd1fee6b Alexis Wilhelm<alexis.wilhelm... 1 Jul 2013 17:29:10
Merge branch 'master' of git.univ-bpclermont.fr:+groupev/libv/develop Bezout<datta.ramadasan@gm... 1 Jul 2013 15:09:26
build: copy headers to build dir instead of linking them Alexis Wilhelm<alexis.wilhelm... 1 Jul 2013 15:01:19
libv_common.cmake : include_directories(BEFORE) Bezout<datta.ramadasan@gm... 1 Jul 2013 15:09:07
Added a bunch of raster image drawing functions Vadim Litvinov<vadim.litvinov... 19 Jun 2013 14:01:39
Merge branch 'master' of ssh://git.univ-bpclermont.fr/+groupev/libv/de... Vadim Litvinov<vadim.litvinov... 17 Jun 2013 17:46:01
frameserver: make boost.chrono optional Alexis Wilhelm<alexis.wilhelm... 14 Jun 2013 10:54:36
display: clear command lists whenever possible Alexis Wilhelm<alexis.wilhelm... 13 Jun 2013 14:02:02
core/config: fix doc Alexis Wilhelm<alexis.wilhelm... 27 May 2013 14:55:52
core/config: load configs from environment Alexis Wilhelm<alexis.wilhelm... 27 May 2013 14:50:16
graphic/palette: fix #include Alexis Wilhelm<alexis.wilhelm... 27 May 2013 14:29:42
graphic/palette: Siméon was not satisfied with my algorithm. Alexis Wilhelm<alexis.wilhelm... 17 May 2013 16:53:22
graphic/palette.hpp: better doc Alexis Wilhelm<alexis.wilhelm... 17 May 2013 14:29:34
frameserver: add FindCImg.cmake Alexis Wilhelm<alexis.wilhelm... 17 May 2013 10:25:00
graphic: add palette.hpp Alexis Wilhelm<alexis.wilhelm... 17 May 2013 10:03:07
core: add path.hpp Alexis Wilhelm<alexis.wilhelm... 15 May 2013 15:29:30
build: account for dependencies between subdirectories Alexis Wilhelm<alexis.wilhelm... 15 May 2013 12:59:23
image_processing: fix harris/zncc
Author Alexis Wilhelm<alexis.wilhelm@univ-bpclermont.fr>
Author date 15 Nov 2013 10:54:07
Parent io/codecs/video/libav.cpp: fix lexical_cast
image_processing: fix harris/zncc
Up
Log
Diff
image_processing/src/libv/image_processing/d
image_processing/src/libv/image_processing/d
image_processing/src/libv/image_processing/d
Loaded 348 revisions (91 KB), time elapsed: 2299 ms (0.04 MB/s)

Git

Installation

Linux

Ubuntu : `sudo apt-get install git`
Debian : `aptitude install git`
Fedora : `yum install git-core`

Mac

```
sudo port install git-core +svn +doc +bash_completion +gitweb
```

Windows

Voir : `http://git-scm.com/download/win`

Création d'un compte

Sites web

- https://gitlab.com/
- https://github.com/
- https://bitbucket.org/
- http://forge.clermont-universite.fr/
- ...

Générer une clef SSH

```
ssh-keygen
cat .ssh/id_rsa.pub
```

Configuration de Git

Nom et adresse

```
git config --global user.name "John Doe"
git config --global user.email johndoe@example.com
```

Couleurs

```
git config --global color.ui true
```

Alias

```
git config --global alias.st status
```

Fichiers à ignorer

```
git config --global core.excludesfile ~/.gitignore
```

Voir le fichier : `~/.gitconfig`

Configuration de Git

Editeur de texte par défaut

Vi est par défaut, pour le remplacer par *emacs* :

```
git config --global core.editor emacs
```

Personnaliser les couleurs

```
git config --global color.diff.meta "blue black bold"
```

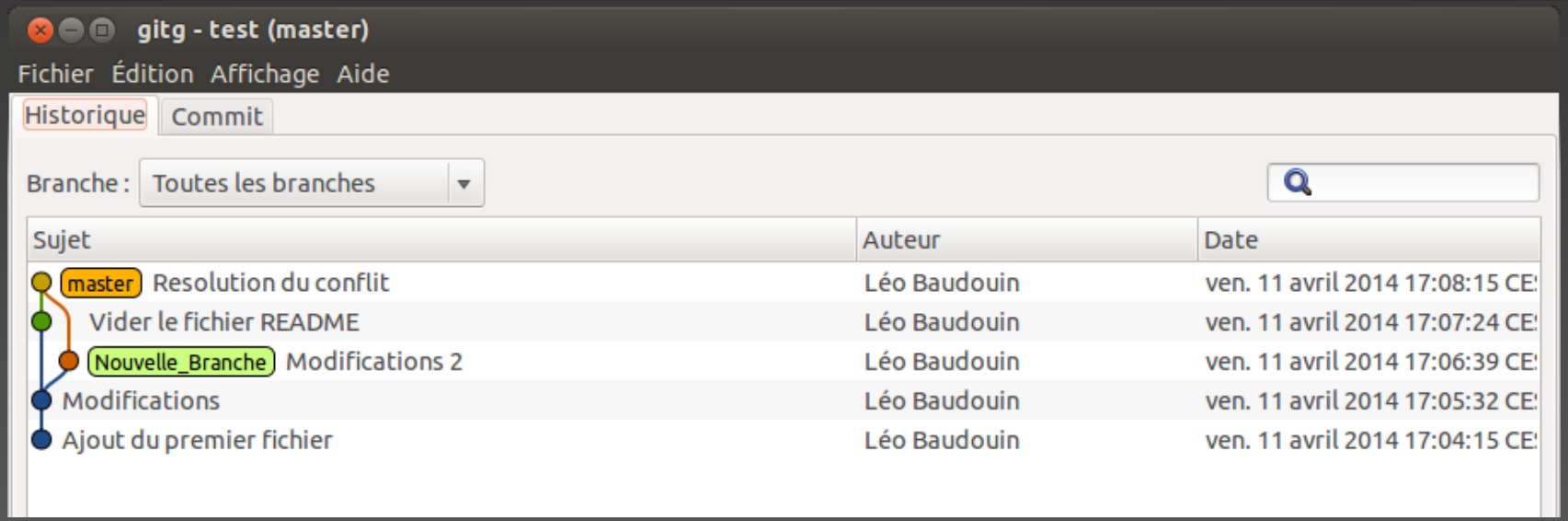
Auto-correction des erreurs de frappe

```
git config --global help.autocorrect 1
```

Git UI

Logiciels utiles

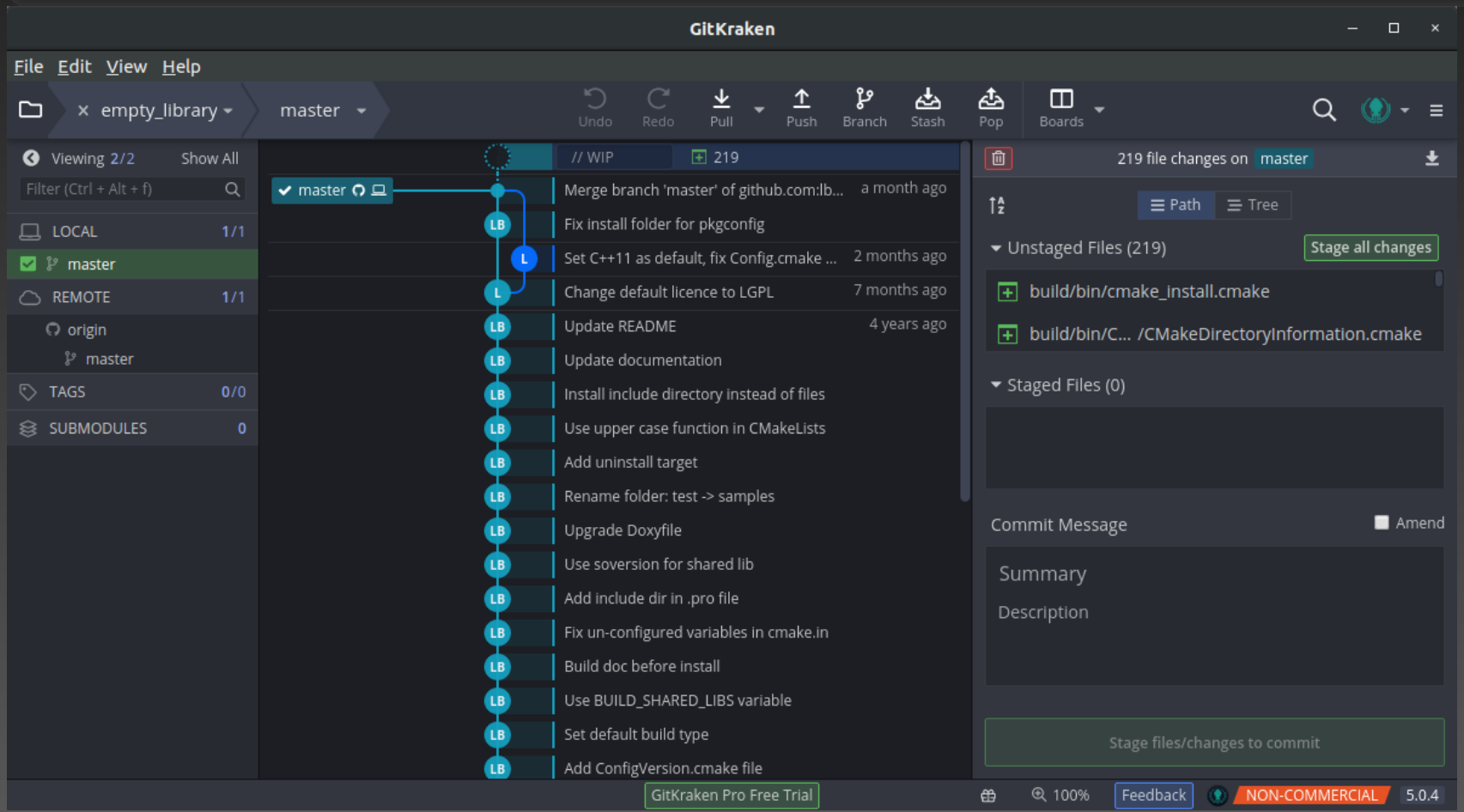
- **gitkraken**
- gitg
- qgit
- gitk
- git gui
- git-cola
- tortoise-git
- ...



Gitkraken

Installation

```
sudo snap install gitkraken
```



Débuter un projet

Depuis un projet existant

```
git clone git@adresse:projet/depot.git
```

Existe en https :

```
git clone https://adresse/projet/depot.git
```

Nouveau projet

```
cd mon_projet
```

Initialiser le dossier :

```
git init
```

Faire le lien avec le serveur :

```
git remote add origin git@adresse:projet/depot.git
```

Exercice

Créer votre premier dépôt

Associer votre clef SSH publique à votre compte Github.
Sur Github, créer un projet "module-git".

Cloner votre premier dépôt

Ouvrir un nouveau terminal :

```
cd Documents/
git clone https://github.com/lbaudouin/module-git.git
```

Utilisation du proxy de l'université

```
git config --global http.proxy
http://user:password@sciproxy.sciences.lan:60158
```

Pour info : contenu du dossier .git

```
lbaudouin@lbaudouin:/tmp/test$ tree .git/
.git/
├── branches
├── COMMIT_EDITMSG
├── config
├── description
├── HEAD
├── hooks
├── index
├── info
│   └── exclude
├── logs
│   ├── HEAD
│   └── refs
│       └── heads
│           └── master
├── objects
│   ├── 0e
│   │   └── 3984ec8da1619fffc948ba3a8fe2a20b37b67e
│   ├── b8
│   │   └── a0d9588093524749853cace8ab6f45d873da5b
│   ├── fb
│   │   └── cf12d50552354fc878706bacea89fbb3f9f999
│   ├── info
│   └── pack
└── refs
    ├── heads
    │   └── master
    └── tags
```

config : fichier relatif à la configuration de l'environnement Git, comme par exemple des informations sur le développeur (son nom, son email ...)

description : contient les informations sur votre projet

objects/ : c'est dans ce répertoire que sont stockés tous les objets Git (commits, tags, trees, blobs)

ref/* : contient les informations sur les branches locales du repository

logs/* : contient les messages de logs

index : fichier contenant des informations sur l'état du prochain commit

HEAD : Pointeur sur la branche actuelle

hooks/ : Dossier contenant des "hooks" ou "triggers", c'est à dire des actions/scripts pouvant être exécutés en pre ou post condition

Ajouter et envoyer un fichier

Créer un nouveau fichier

```
echo "Mon projet" > README
```

Ajouter le fichier à la liste des fichiers suivis

```
git add README
```

Créer un commit avec les fichiers modifiés

```
git commit -m "Création du fichier README"
```

Envoyer le commit sur le serveur

```
git push
```

Récupérer les modifications

Récupérer les modifications sur le serveur

```
git pull
```

Explication

`git pull = git fetch + git merge`[a]

`git fetch` : Récupère les données sur serveur

`git merge` : Fusionne avec le dépôt local

---

a. On peut remplacer `git merge` par `git rebase` (voir explication dans la suite) avec `git pull --rebase` ou de façon permanente avec `git config --global pull.rebase true`

Exercice

Créer et modifier un dépôt

Créer un dépôt sur un des sites

```
mkdir <dossier_du_projet>
cd <dossier_du_projet>
git init
git remote add origin git@github.com:<depot>.git

git add <fichiers>
git commit -m "<Description>"
git push
```

Voir les modifications

Liste des fichiers locaux modifiés

```
git status
```

Liste des modifications apportées

```
git diff
git diff <mon_du_fichier>
git diff <commit1> <commit2>
```

Afficher les logs

```
git log [--stat] [-<n>]
```

Modifications ligne par ligne

```
git blame <nom_du_fichier>
```

Exercice

Modifier le dépôt d'un autre étudiant

Ajouter un étudiant en tant que collaborateur sur le serveur en utilisant l'interface web.

Etudiant 1 :

```
git clone git@adresse:depot.git

git add <fichiers>
git commit -m "Description"
git push
```

Etudiant 2 :

```
git pull
git log
```

Gestion des conflits

Visualiser les conflits

```
<<<<<<<<<<<<<
V1
=============
V2
>>>>>>>>>>>>>
```

Mettre à jour les fichiers

```
git add <nom_du_fichier_en_conflit>
```

Exercice

Gestion des conflits

Etudiant 1 & 2 :

```
git pull
```

Etudiant 1 :

Modifier un fichier

```
git add <fichier>
git commit -m "Description"
git push
```

Etudiant 2 :

Modifier les mêmes lignes du même fichier

```
git add <fichier>
git commit -m "Description"
git push
```

Tips

Raccourcis pratiques

- `git add -u`
  Ajoute/supprime tout les fichiers suivis modifier/supprimer
- `git commit -a -m "Description"`
  = `git add -u` + `git commit -m "Description"`
- `git difftool -t <prog> <commit1> <commit2> <file>`
  Diff avec un programme externe : meld, kompare, ...
- `git rm <fichier>`
  Retire le fichier de la liste des fichiers suivis

Aide sur git

Pour obtenir de l'aide sur une fonction :

`man git-<fonction>`

Exemple : `man git-commit`

Gestion des branches

Afficher les branches

`git branch [-a]`

ou `git show-branch`

Créer une nouvelle branche

`git branch <ma_branche>`

ou `git checkout -b <ma_branche>`

Changer de branche

`git checkout <nom_de_la_branche>`

Fusionner les branches

`git checkout <branche_principale>`

puis `git merge <branche_secondaire>`

Gestion des branches

Créer une branche sur le serveur

```
git push <nom_serveur> <nom_de_la_branche_locale>
```

Forcer l'association d'une branche locale et d'une branche distante

```
git branch -f <br_locale> <nom_serveur>/<br_distante>
```

Supprimer une branche locale

```
git branch -d <nom_de_la_branche_locale>
```

Supprimer une branche sur le serveur

```
git push <nom_serveur> :<nom_de_la_branche_locale>
```

Exercice

Gestion des branches

- Travailler sur une nouvelle banche pendant qu'un autre étudiant travaille sur la branche principale.
- Fusionner la nouvelle branche avec la branche principale
- Regarder le résultat avec gitg

Gestion des tags

Créer un tag

Après avoir fait le commit :

```
git tag -a "<nom du tag>" -m "<description du tag>"
```

Envoyer le tag

```
git push --tags
```

Revenir à un certain tag

```
git checkout "<nom du tag>"
```

Mettre en attente (remiser)

Mettre en attente des modifications

```
git stash
```

Lister les mises en attente

```
git stash list
```

Récupérer les modifications en attente

```
git stash pop
```

Supprimer les modifications en attente

```
git stash drop
```

Créer une branche à partir des modifications en attente

```
git stash branch <nom_branche>
```

Autres fonctions

Modifier le dernier commit (non pushé)

`git commit --amend`

Debug par recherche dichotomique

`git bisect`

Voir `man git-bisect`

Utiliser des sous-modules

Utiliser des dépôts Git dans un dépôt Git, par exemple pour des sous-parties facultatives d'un programme :

`git submodule`

Autres fonctions

Commandes

- `git reset`
  Retourne à un état précédent (option --hard)
- `git revert`
  Annule les modifications d'un commit en en générant un nouveau avec les modifications inverses
- `git rebase`
  Applique les modifications à la suite au lieu d'effectuer un *merge*

Exercices en lignes

Exercices sur Git

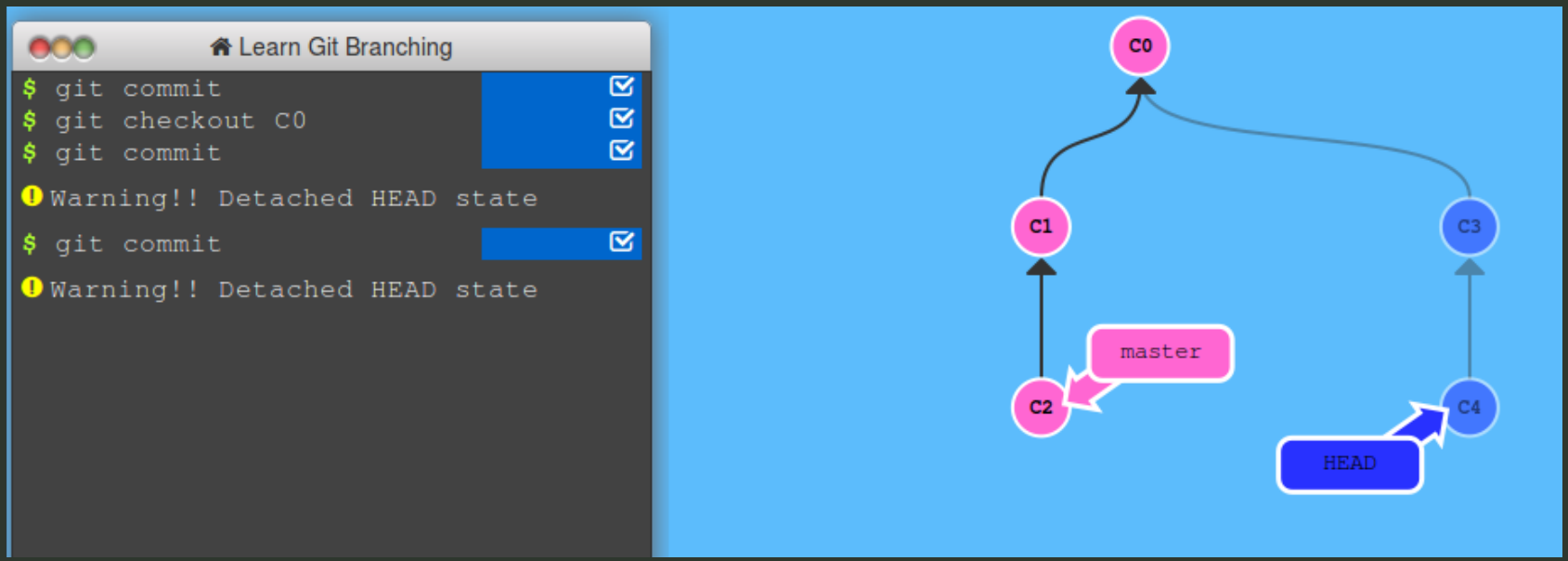
`http://pcottle.github.io/learnGitBranching/`

Dépôt virtuel

`http://pcottle.github.io/learnGitBranching/?NODEMO`

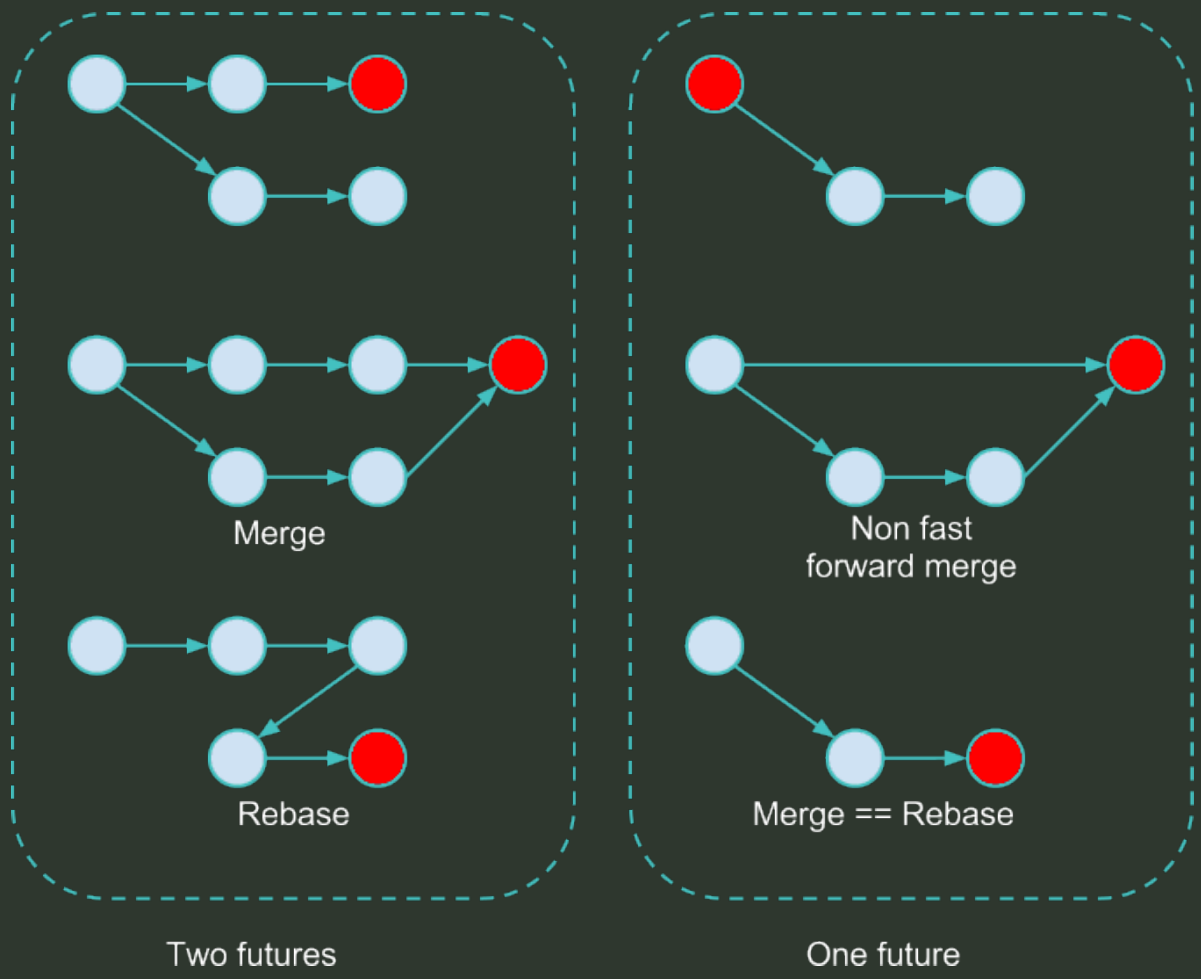
Exercice

Différence entre **merge** et **rebase**

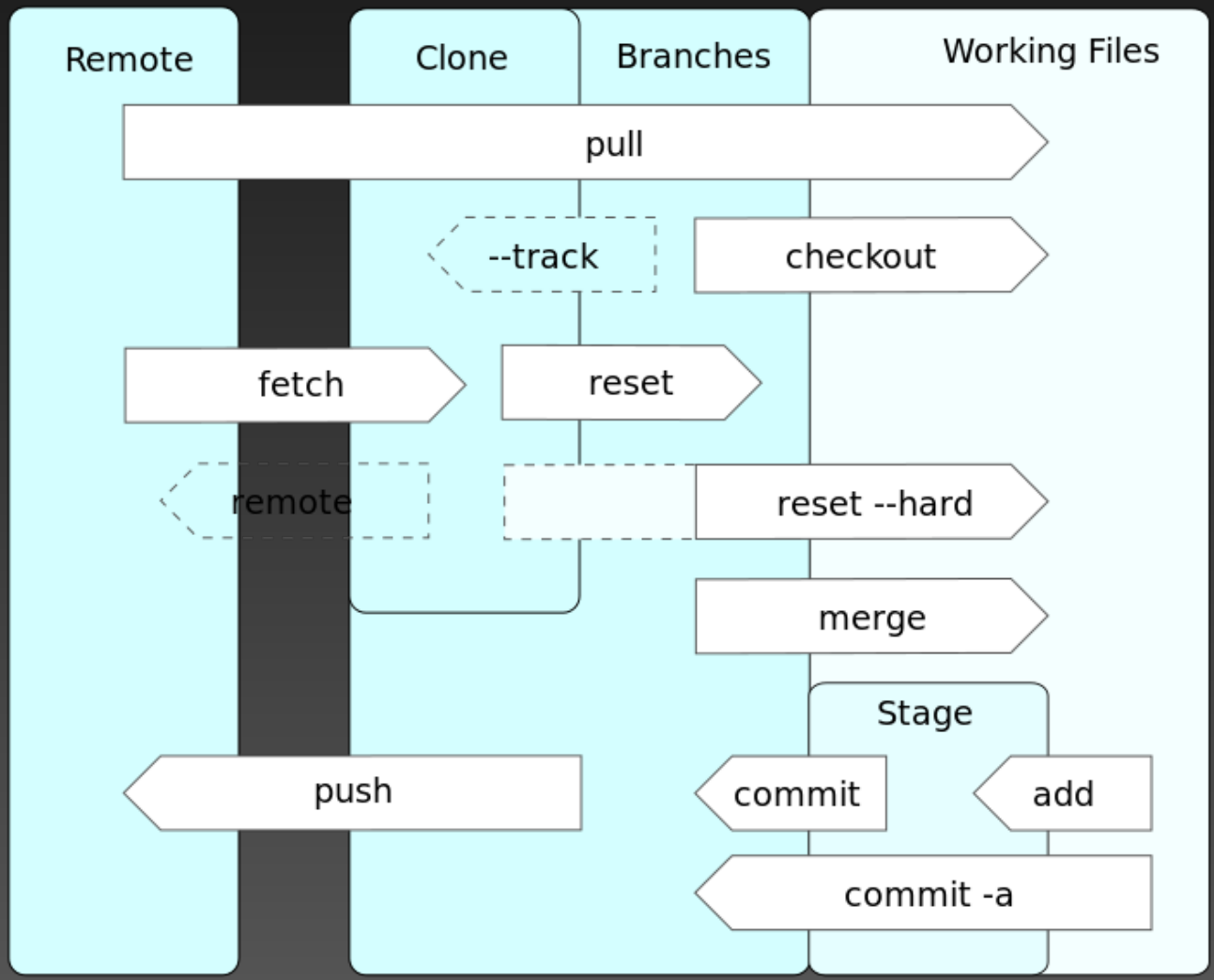


Exercice

Différence entre **merge** et **rebase**



Rappel


Remote
Clone
Branches
Working Files
pull
--track
checkout
fetch
reset
remote
reset --hard
merge
Stage
push
commit
add
commit -a