Suivez cette présentation sur votre ordinateur :

`https://louvainlinux.org/atelier-git`

Préparez-vous à utiliser `git` : vous utiliserez le logiciel GitHub Desktop durant cette présentation.

Prenez un peu d'avance, créez-vous un compte Github et installez Github Desktop :

- Sur les ordinateur Windows UCL : installez `https://desktop.github.com`
- Ou installez GitHub Desktop sur votre ordinateur :
  - Ubuntu : `https://github.com/shiftkey/desktop/releases`
  - Windows ou OS X : `https://desktop.github.com`

Présentation Git

Un outil de collaboration puissant

Théo Vanden Driessche Morgan Leclerc

10 Octobre 2019

KAP Louvain-li-Nux

Cette présentation

- Cette présentation est sous license libre CC-BY 4.0.
- En ligne (slides en pdf et sources LaTeX, exercices…) : https://github.com/louvainlinux/atelier-git

Table des matières

1. Introduction

2. Principes de Git

3. Utilisation : en pratique

4. Exercices

5. Fonctionnalités plus avancées

6. Informations et ressources

Introduction

Gérer un projet

Comment gérez-vous actuellement un projet ?

- L'envoyer à travers un message sur Facebook, ... (**Très mauvaise idée**)
- L'envoyer par mail (**Un peu moins**)
- Utiliser une Dropbox, Google Drive, ... (**Déjà mieux mais toujours risqué ou manque de fonctionnalités**)

Solution : Utiliser un **système de gestion de version décentralisé** (Distributed Version Control System (DVCS) pour les anglophiles).

Un DVCS ?

- **Version** Enregistre des « instantanés » du projet.
- **Gestion** Revenir en arrière, voir des différences, fusionner des modifications.
- **Décentralisé** Chacun
  - a sa copie (avec son historique) sur son PC,
  - peut mettre sa copie (et son historique) en ligne,
  - peut récupérer sur son PC les copies et historiques disponibles en ligne,
  - peut fusionner différentes copies (semi-)automatiquement.
- **Projet** n'importe quel répertoire (« dossier ») sur votre ordinateur. Donc n'importe quoi : Bureautique, LaTeX, code, images, musique…

Et Git dans tout ça ?

`Git` a été créé en 2005 par Linus Torvalds (auteur de `Linux`); le plus connu et utilisé.

À l'origine, interface en ligne de commande.

Aujourd'hui : aussi des interfaces graphiques, dont GitHub Desktop.

Mais on m’avait parlé de GitHub !

Souvenez-vous...

- **Décentralisé** Chacun
  - peut mettre sa copie (et son historique) en ligne,
  - ...

Il y a plein d’”endroits” en ligne où on peut envoyer son travail, GitHub est le plus connu.

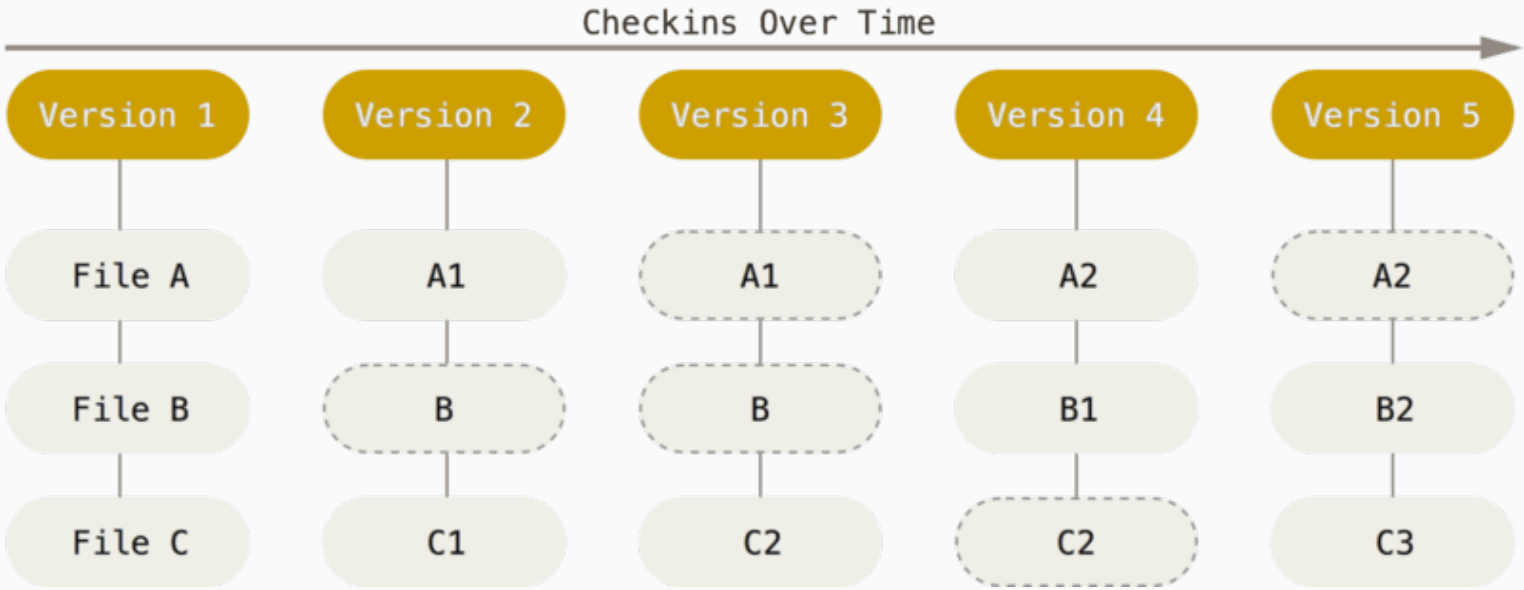
En plus de ça, GitHub a des fonctionnalités pour interagir avec des collaborateurs.

Principes de Git

Concepts

- **Espace de travail :** les fichiers, répertoires... dans lesquels on travaille. Ils n'ont rien de spécial par rapport à d'autres dossiers sur l'ordinateur.
- **Dépôt :** espace de travail + historique, sur un ordinateur.
- **Commit :** "version", est le successeur d'une autre commit.
- **Historique :** la "chaîne" de tous les commits, du plus ancien au plus récent.
- **Dépôt distant :** un dépot qui se trouve chez GitHub.

Concept : le commit



Les illustrations non-sourcées viennent de `https://git-scm.com/book`.

Actions

- **Créer** un dépot sur GitHub.
- **Cloner** (faire une copie d') un dépot de GitHub sur son PC.
- **Modifier/créer** des fichiers (pas avec Git !).
- **Ajouter** un fichier modifié : il sera pris en compte dans le prochain commit.
- **Faire un commit** : créer une nouvelle version, qui contient tous les fichiers ajoutés. On y ajoute un commentaire (qui décrit les changements).

Action

- **Consulter** un historique.
- **Push** : envoyer ses nouveaux commits sur GitHub.
- **Pull** : récupérer des changements (qui ont été envoyés par quelqu'un d'autre) depuis GitHub.
- **Merge** : quand on Pull et qu'on a aussi des nouveaux commits sur son PC. Git essaye de fusionner automatiquement ; s'il ne sait pas le faire, il demande à l'utilisateur.

Questions?

Utilisation : en pratique

Commandes

Définir son profil :

```
git config --global user.name « nom-complet»
git config --global user.email <email>
git config --global user.editor <text-editor>
```

Créer un dépot local :

```
git init
```

Ajouter un dépot distant :

```
git remote add <remote-url>
```

Voir l'état du repos et les fichiers ajoutés :

```
git status
```

Ajouter des fichiers :

```
git add <fichiers>
```

Commandes

Commit les changements ajoutés :

`git commit`

Visualisez l'historique :

`git log`

Ignorer un fichier : Ajouter le path du fichier dans `.gitignore`

Récupérer des commits du repo distant :

`git pull`

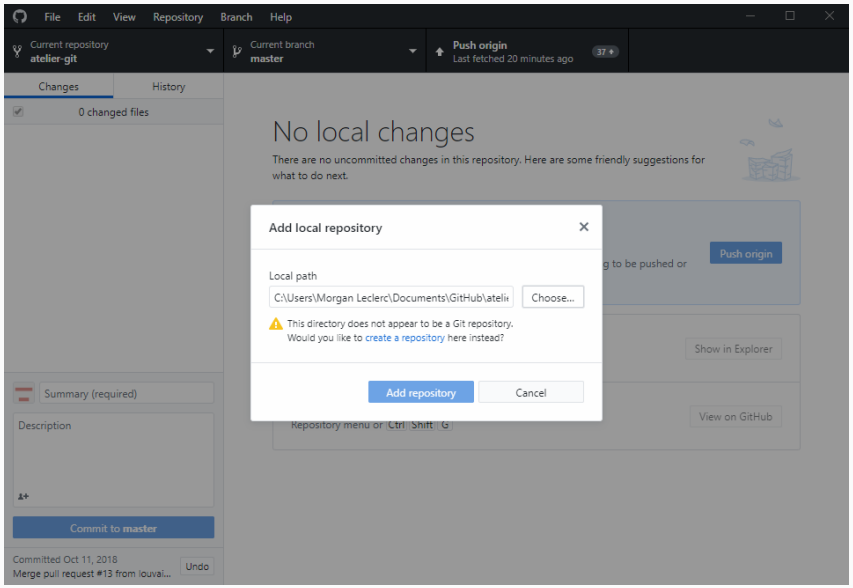
Envoyer des commits au repo distant :

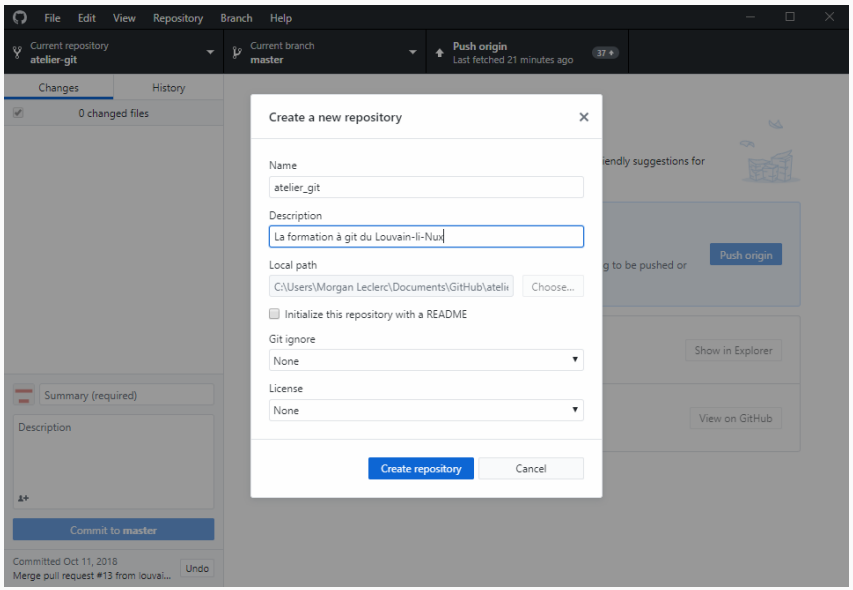
`git push`

Merge une branche dans la branche actuelle :

`git merge <branch>`

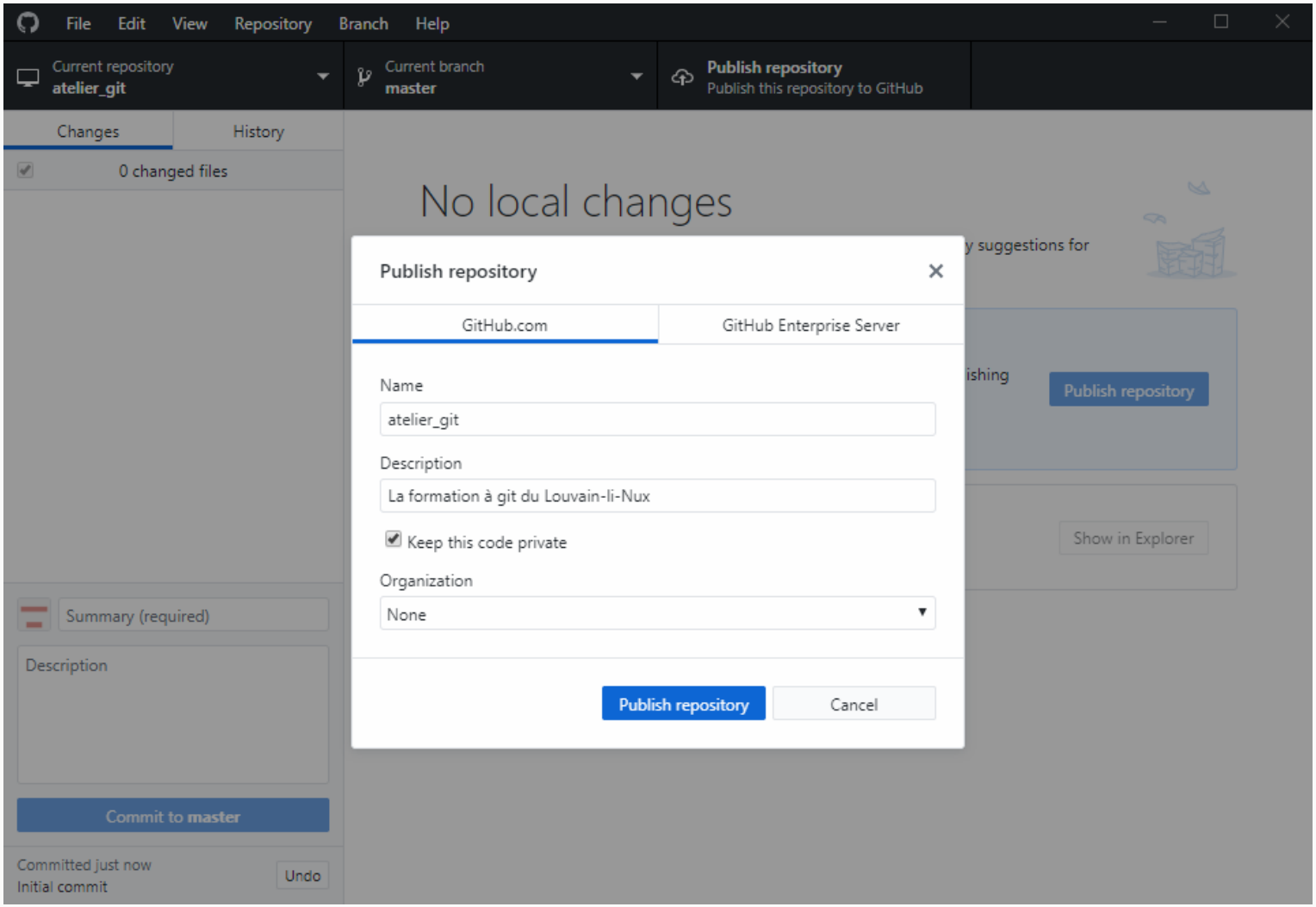
Créer un nouveau dépot local





File → Add Local repository *Ctrl+O*

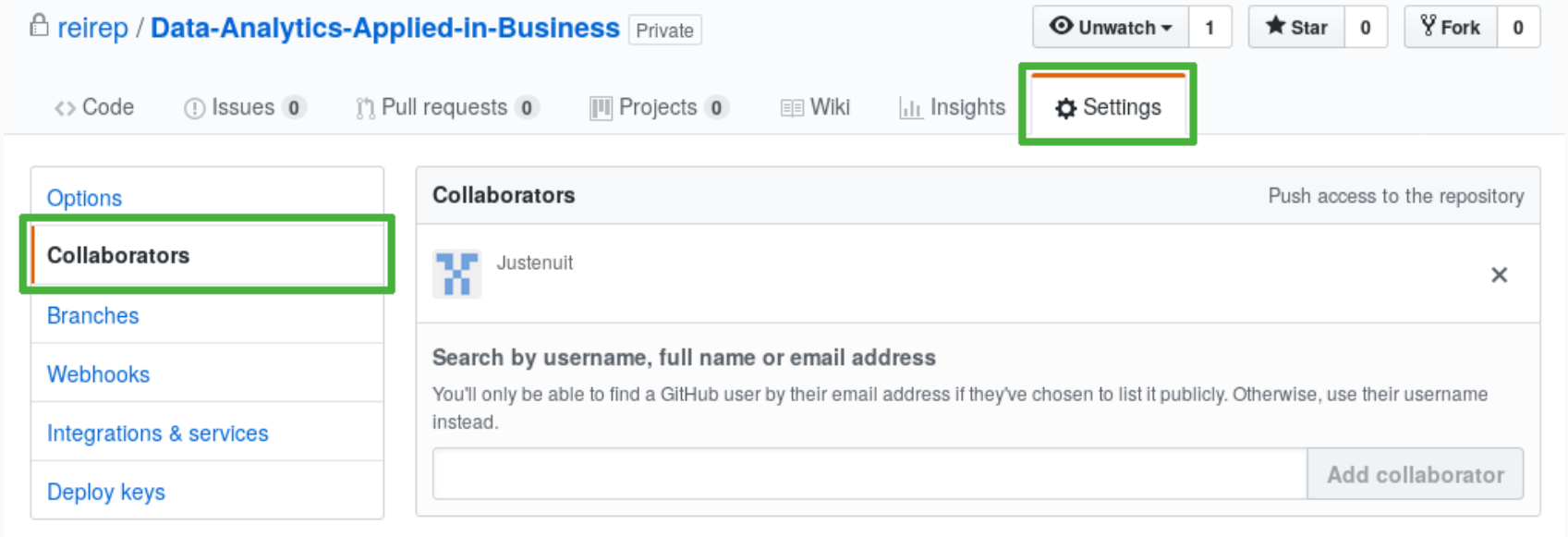
Publier un dépot sur Github



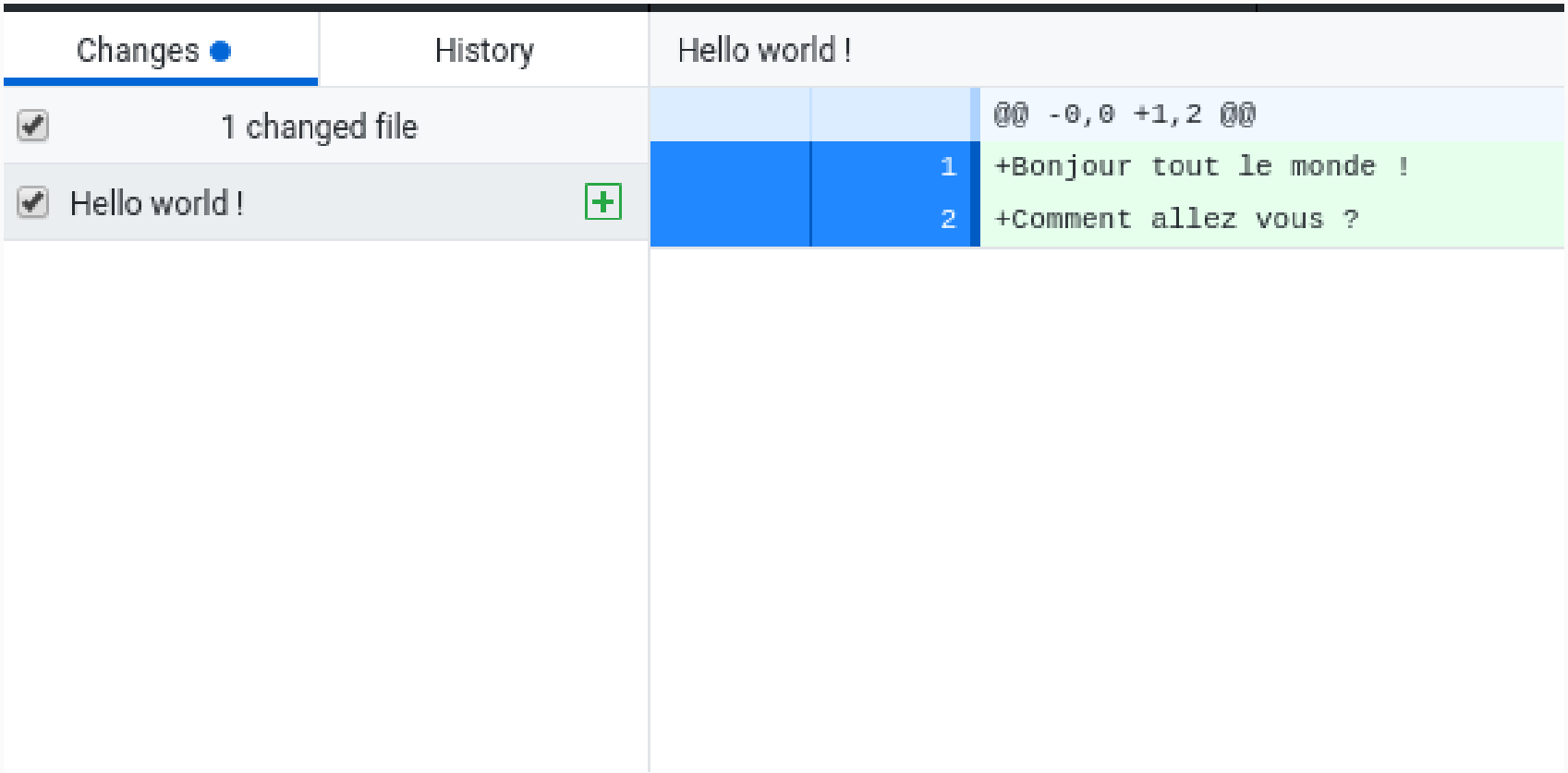
Pour accéder au dépot en ligne : Repository → View on Github

Ctrl+Shift+G

Ajouter un collaborateur sur GitHub

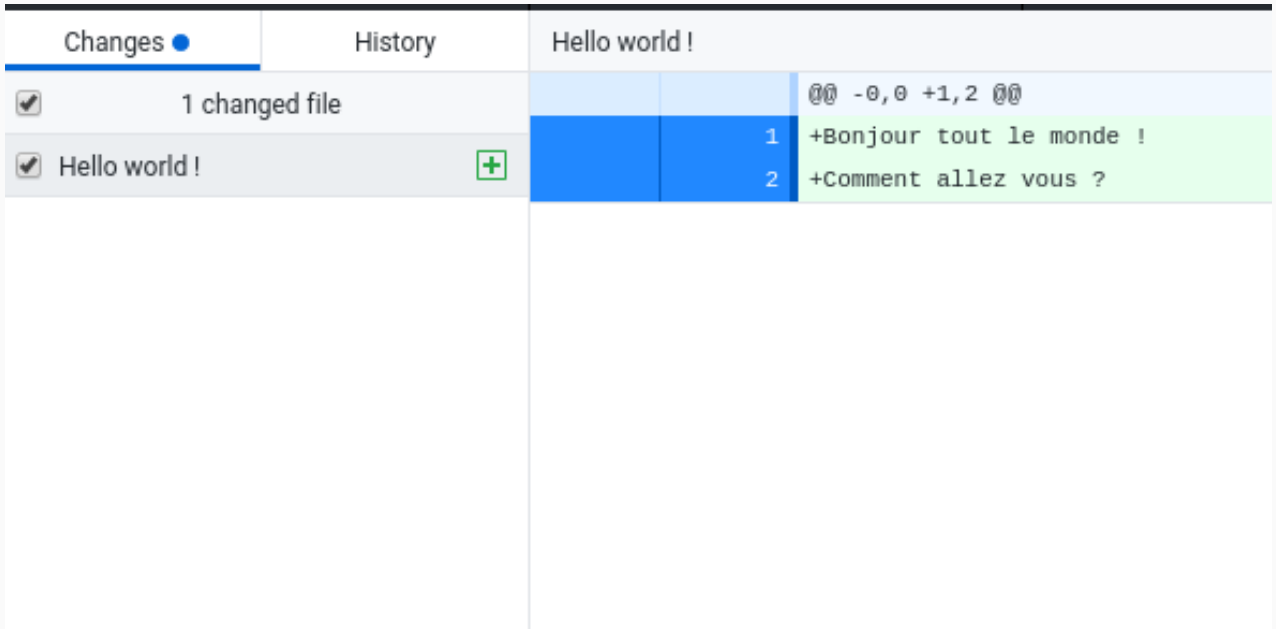


Ajouter des fichiers



Remarque : fichier texte vs binaire

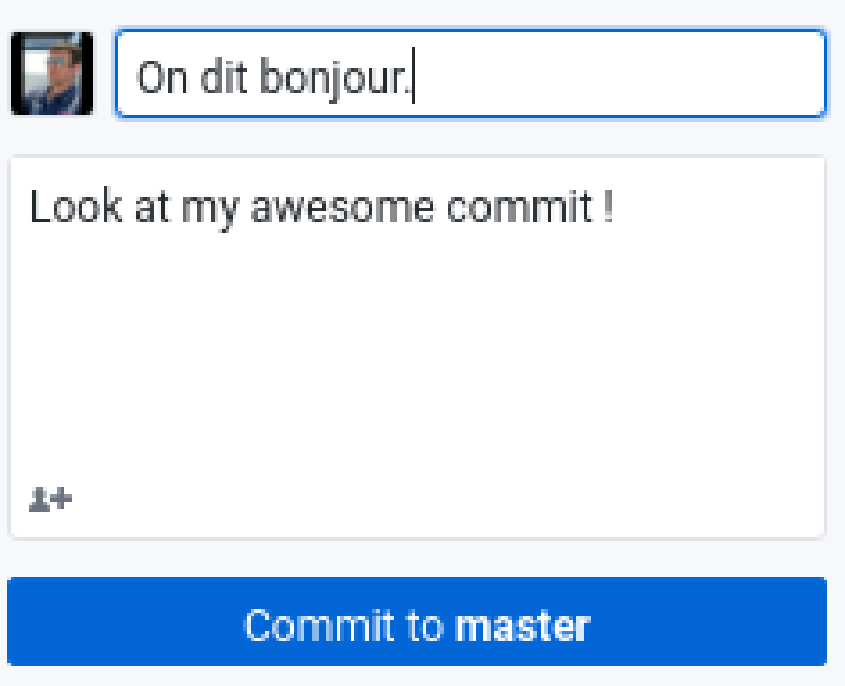
- Fichiers texte : programme, LaTeX...



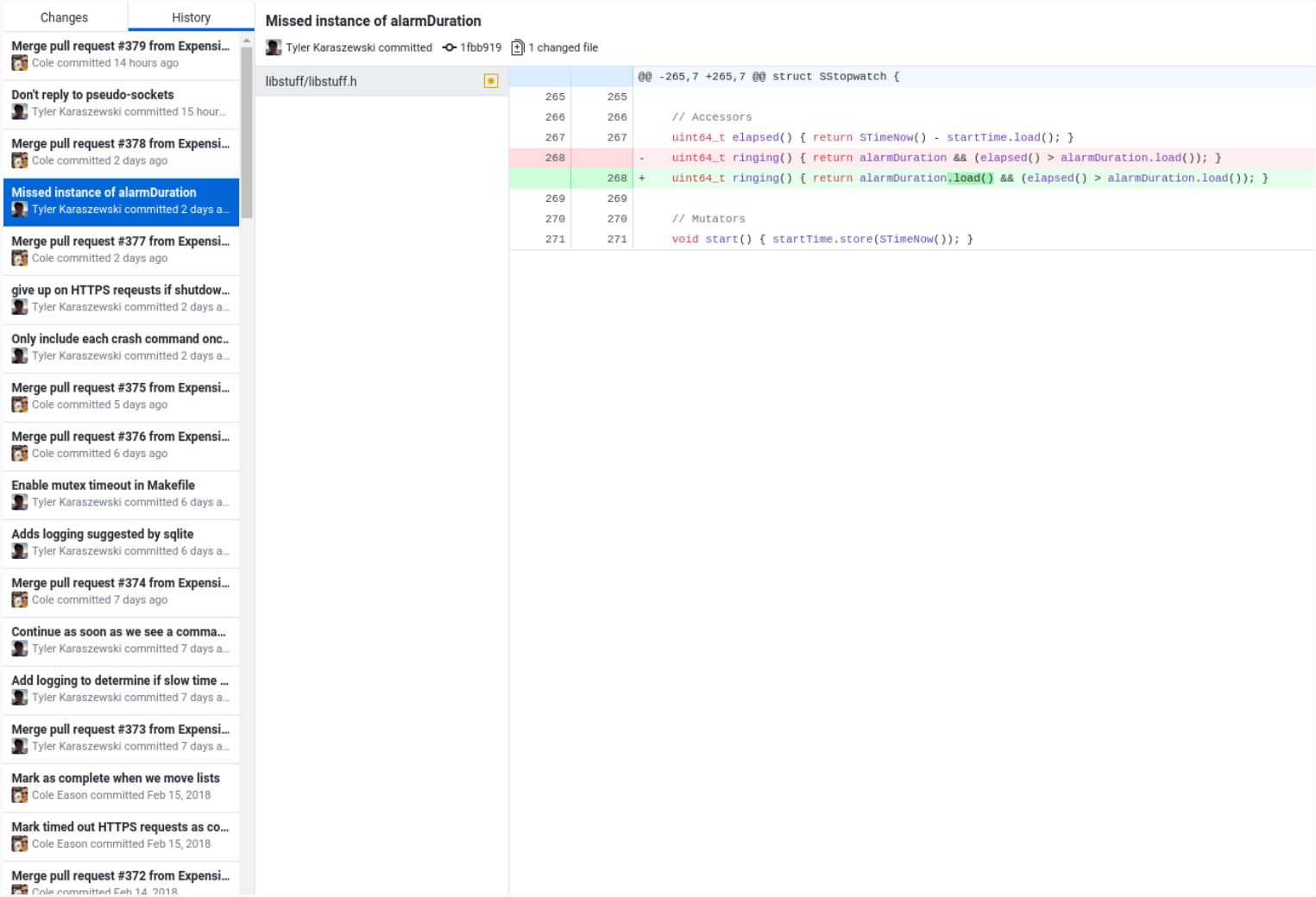
- Fichiers binaires : le reste : Word, Writer, images, sons, PDF...

Créer un commit

- Créer un `commit` sur base des fichiers ajoutés.
- Message de `commit` : décrit les changements effectués.



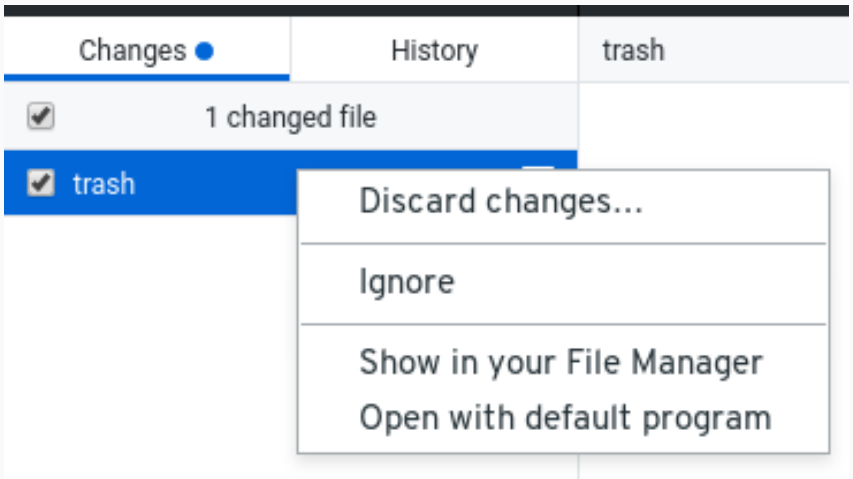
Visualiser l’historique



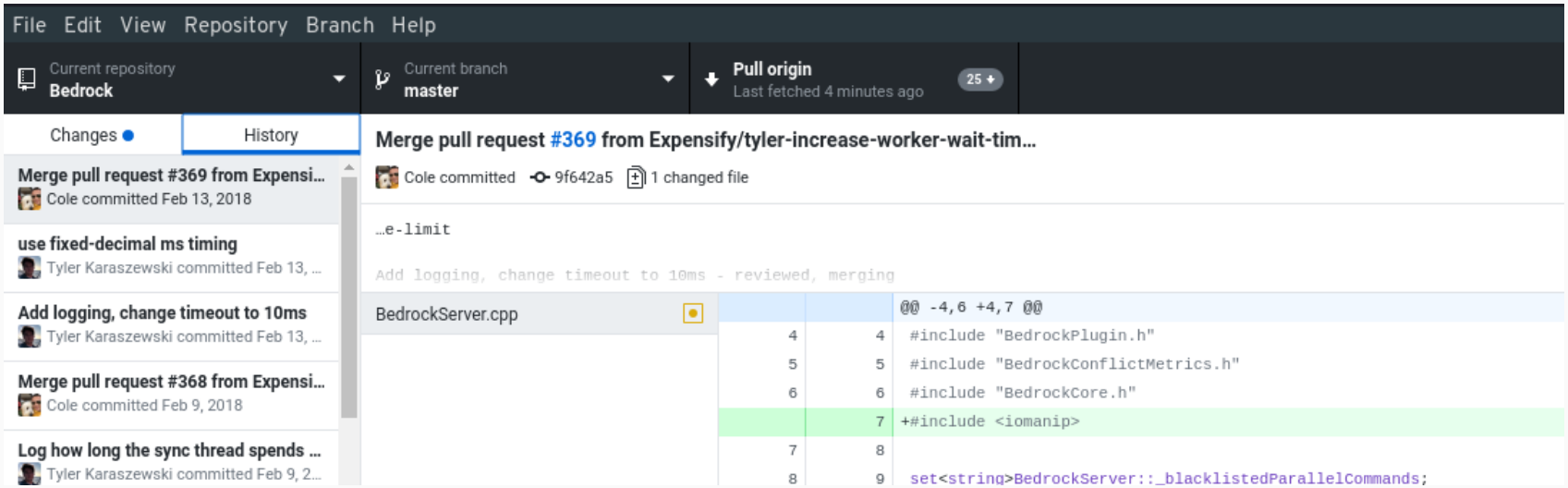
Astuce : ignorer des fichiers

Des fichiers que vous ne voulez jamais dans Git (résultats de compilation, fichiers temporaires…) ? Cachez-les !

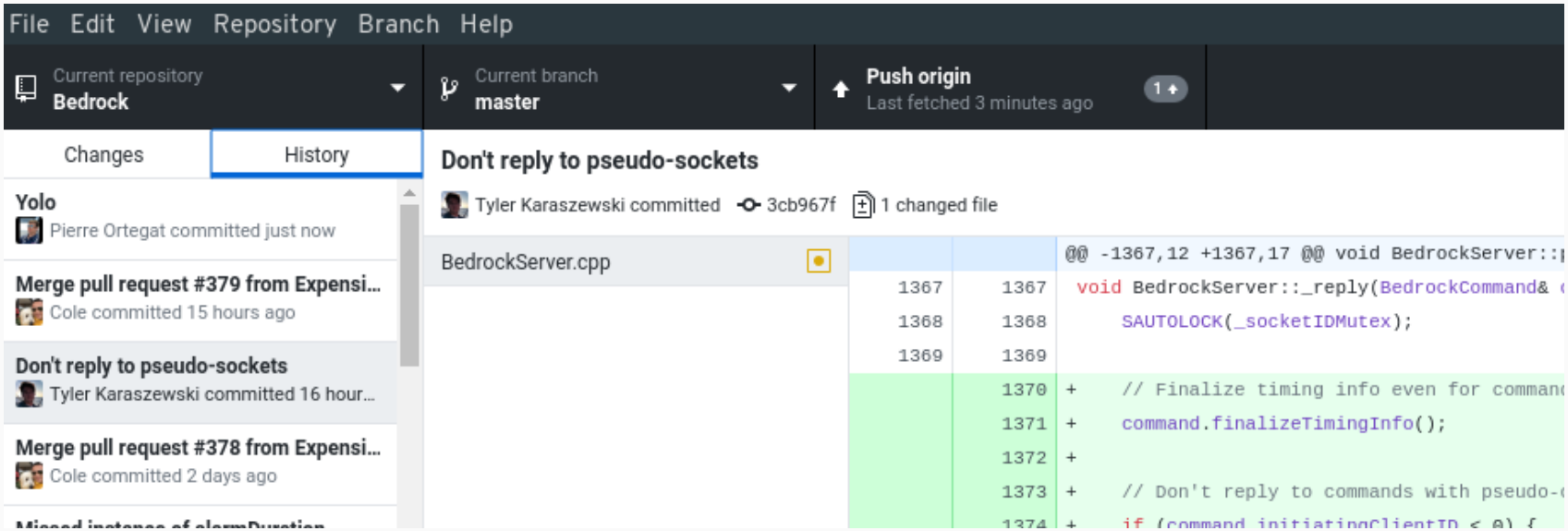
NB : Cela crée un fichier `.gitignore` : celui-là, on le versionne.



Pull : récupérer des commits qui sont sur GitHub

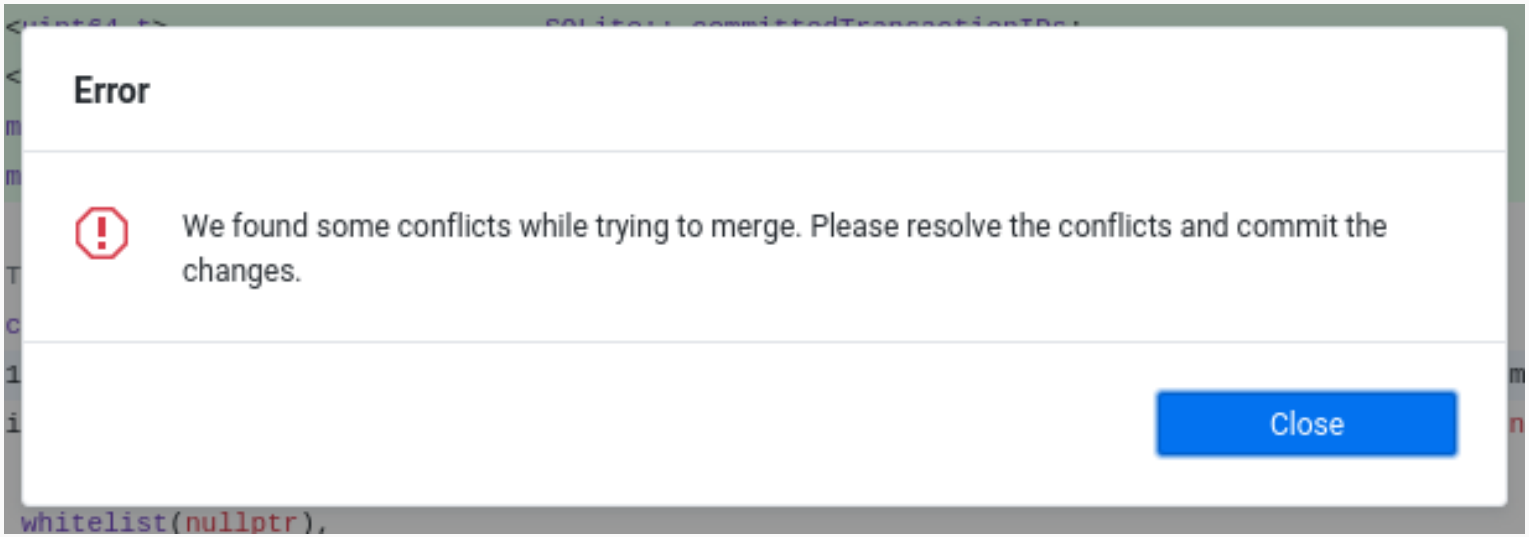


Push : envoyer des commits sur GitHub



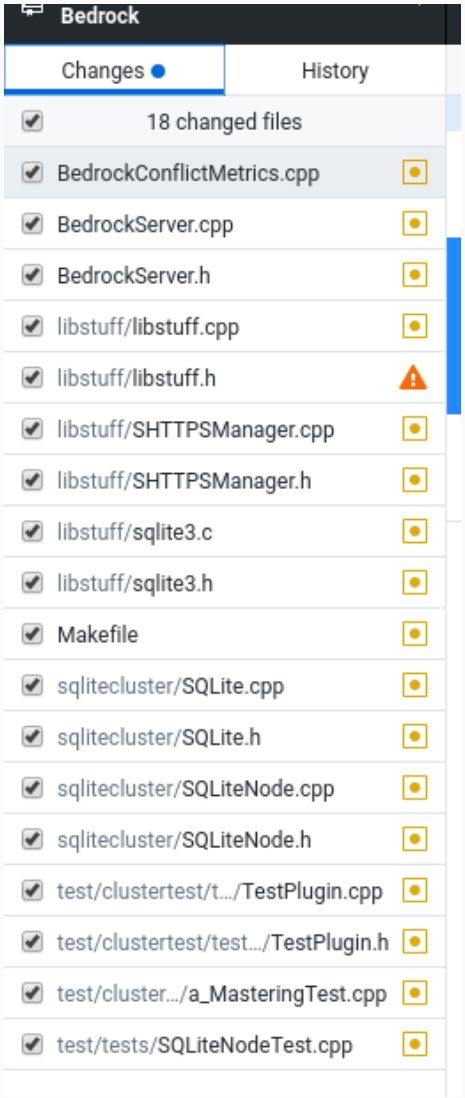
Merge non-automatique : quand il y a des conflits

Message d’erreur :



Merge non-automatique : quand il y a des conflits

Trouver le(s) fichier(s) en conflit :



Merge non-automatique : quand il y a des conflits

Trouver le(s) endroit(s) en conflit dans le fichier (reconnaissables par des balises) :

Avant :

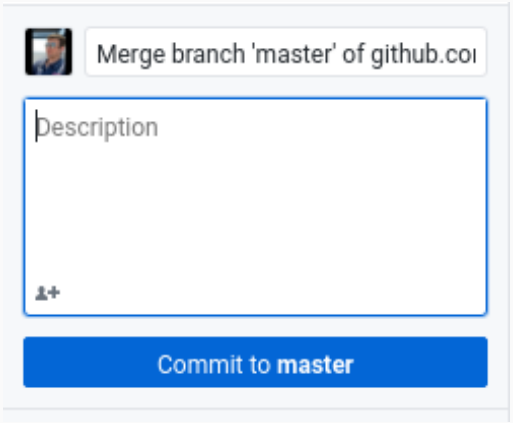
```
    // Accessors
<<<<<<<<<< HEAD
    uint64_t elapsed() { return STimeNow() - startTime.load(); }
    uint64_t ringing() { return alarmDuration.unload() && (elapsed() > alarmDuration.load()); }
==========
    uint64_t elapsed() { return STimeNow() - startTime.load(); }
    uint64_t ringing() { return alarmDuration.load() && (elapsed() > alarmDuration.load()); }
>>>>>>>>>> 3cb967f1fc1b9421cad2f33dce81f258cdeea2b5
```

Après :

```
    // Accessors
    uint64_t elapsed() { return STimeNow() - startTime.load(); }
    uint64_t ringing() { return alarmDuration.load() && (elapsed() > alarmDuration.load()); }
```

Merge non-automatique : quand il y a des conflits

Choisir la version que l'on veut garder et commit :



Astuce : de l'aide !

On peut trouver de l'aide :

Github help :
https://help.github.com/

Github desktop help :
https://help.github.com/desktop/

Exercices

Exercice 1 : partie 1

Exercice à faire par **groupe de 2 ou 3** :

Pour tout le monde :

- Créez un compte sur GitHub.
- Installez GitHub Desktop, (voir liens slide 1), puis ouvrez-le.

Une personne du groupe :

- Créez un dépôt nommé *blagues* sur votre oridnateur.

Exercice 1 : partie 2

Une personne :

1. Créez un fichier avec NotePad++ (ou autre) dans le dépôt sur votre ordinateur.[1]
2. Remplissez le fichier avec des blagues. (Si vous n'avez pas d'idées, cliquez ici)
3. Sauvez le fichier .txt → **NE PAS OUBLIER !!!**
4. Reprenez GitHub Desktop et ajoutez le fichier.
5. Faites un commit.
6. Faites un push vers le dépôt GitHub en ligne.
7. Publiez votre dépot et ajoutez les autres en collaborateurs sur le dépôt. Ils reçoivent une invitation par mail, ils doivent l'accepter.[2]

1. Emplacement par défaut sur les PCs UCL : Z :\GitHub\blagues.
2. Si vous arrivez sur une page 404, connectez-vous à github et ré-essayez.

Exercice 1 : partie 2

Ensuite, les autres, chacun à son tour :

- Faites un pull
- Regardez l'historique pour vérifier que les changements sont là.
- Ajoutez des blagues dans le fichier.
- Refaites les étapes 3 à 6 ci-dessus.

N'hésitez pas à répéter cela plusieurs fois, pour être sûrs de bien comprendre !

Exercice 2 : partie individuelle

Partie à faire individuellement (chacun sur son ordinateur, simultanément) :

- Ouvrez GitHub Desktop et clonez le dépôt précedemment créé si ce n'est déja fait.
- Ajoutez une blague dans le fichiet .txt. Ne pas oublier de sauver le fichier.
- Ajoutez le fichier dans GitHub Desktop.
- Faites un commit. (pas de push)
- Observez et comparez les historiques de chacun.
- Suivez les slides suivants en fonction du nombre personnes dans votre groupe.

Exercice 2 : pour groupe de 2

- Une personne fait un push. L'autre personne ne fait rien.
- L'autre personne fait le pull sur son ordinateur (cliquez sur "push" si "pull" n'est pas affiché, et cliquez "close" sur le message d'erreur qui s'affiche).
- Résolvez le conflit de merge ensemble pour avoir les deux blagues (il faut éditer le fichier en qestion).
- Une fois le merge terminé, faites un commit, puis un push.
- L'autre personne peut faire un pull pour récupérer la dernière blague.
- Comparez à nouveau vos historiques.

Exercice 2 : pour groupe de 3

- Une personne fait un push. Les autres personnes ne font rien.
- Une des 2 autres personnes fait le pull sur son ordinateur (cliquez sur ”push” si ”pull” n’est pas affiché, et cliquez ”close” sur le message d’erreur qui s’affiche).
- Résolvez le conflit de merge ensemble pour avoir les deux blagues (il faut éditer le fichier).
- Une fois le merge terminé, faites un commit et un push.
- La dernière personne fait le pull sur son ordinateur.
- Résolvez le conflit de merge ensemble pour avoir toutes les blagues.
- Les autres peuvent faire un pull pour récupérer toutes les blagues.
- Comparez à nouveau vos historiques.

Exercice 3

Exercice à faire par **groupe de 2 ou 3** :

Pour une personne :

- Créez un fichier avec Word (.docx) dans le dépôt *"blagues"* précédemment créé sur votre ordinateur.
- Remplir le fichier avec des blagues.
- Sauver le fichier .docx.
- Reprendre GitHub Desktop et faire le add du fichier et le commit.
- Faites un push sur le dépôt GitHub en ligne.

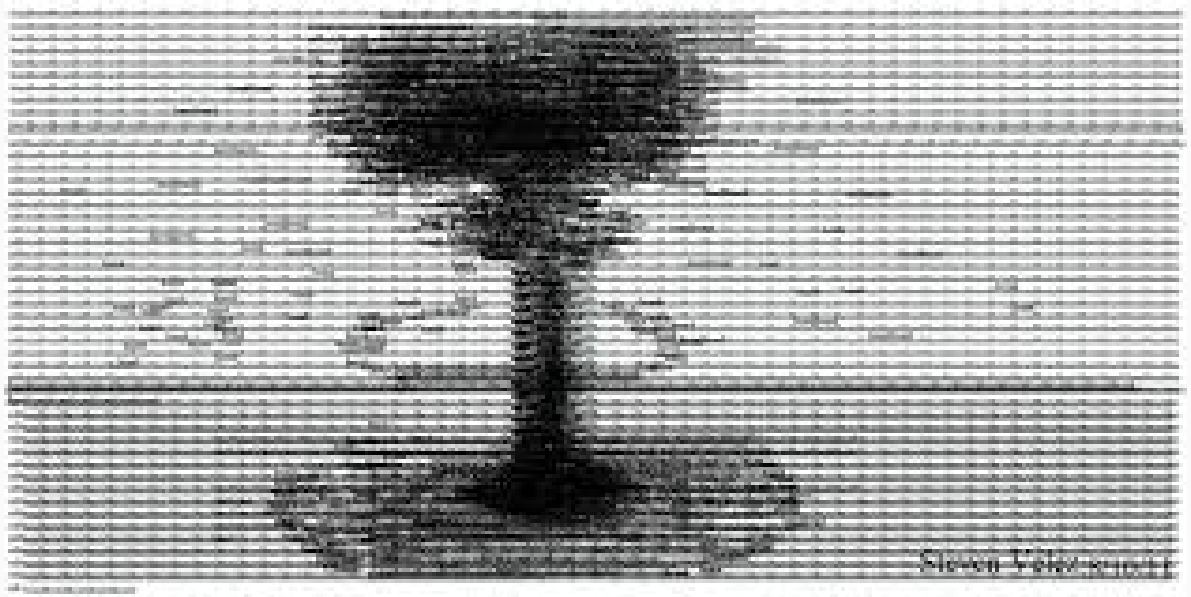
Exercice 3 : partie seul

Pour chacun :

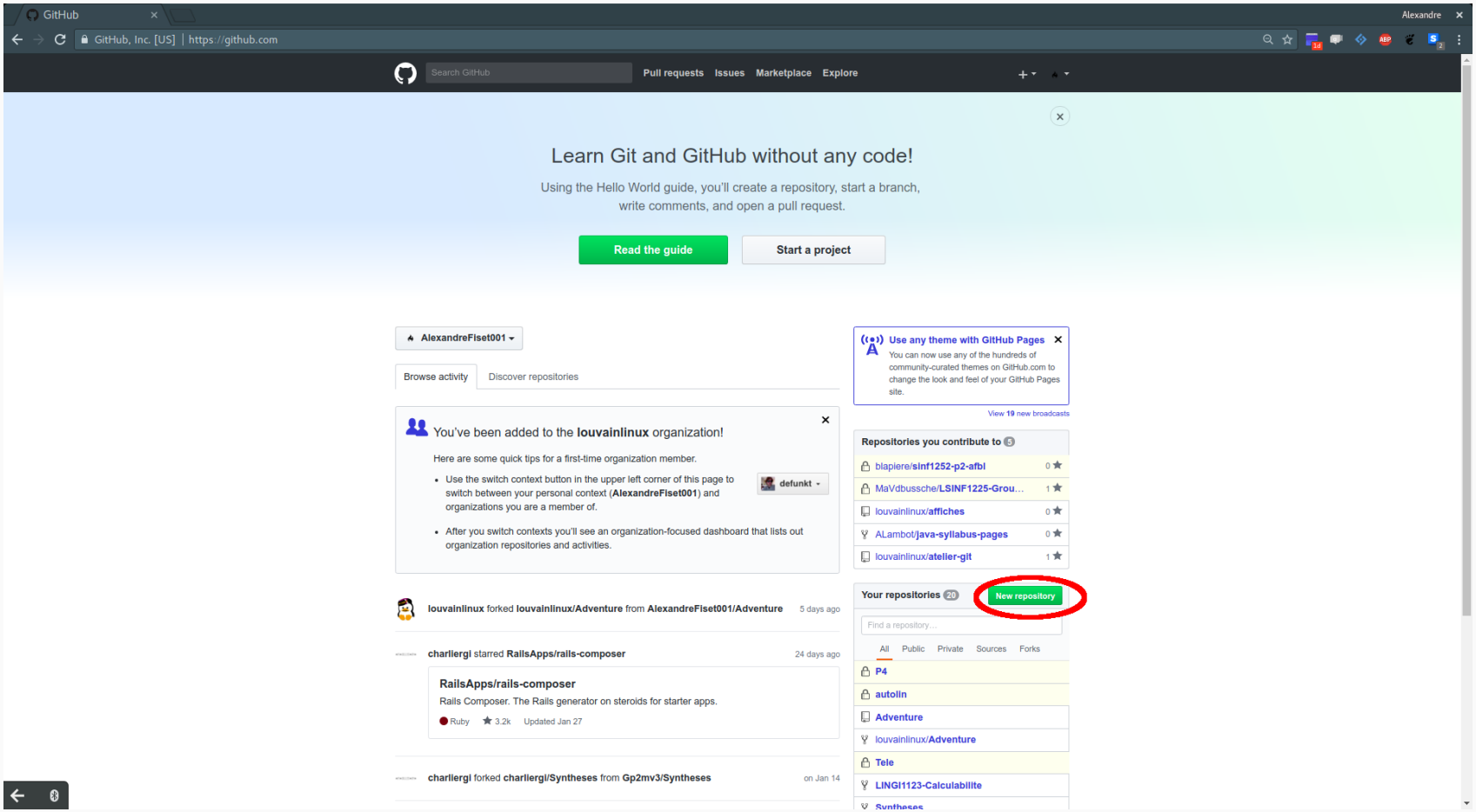
- Faire le pull du dépôt sur votre ordinateur.
- Ajouter une blague au fichier .docx.
- Sauver le fichier .docx
- Faites un commit.

Exercice 3 : partie fun

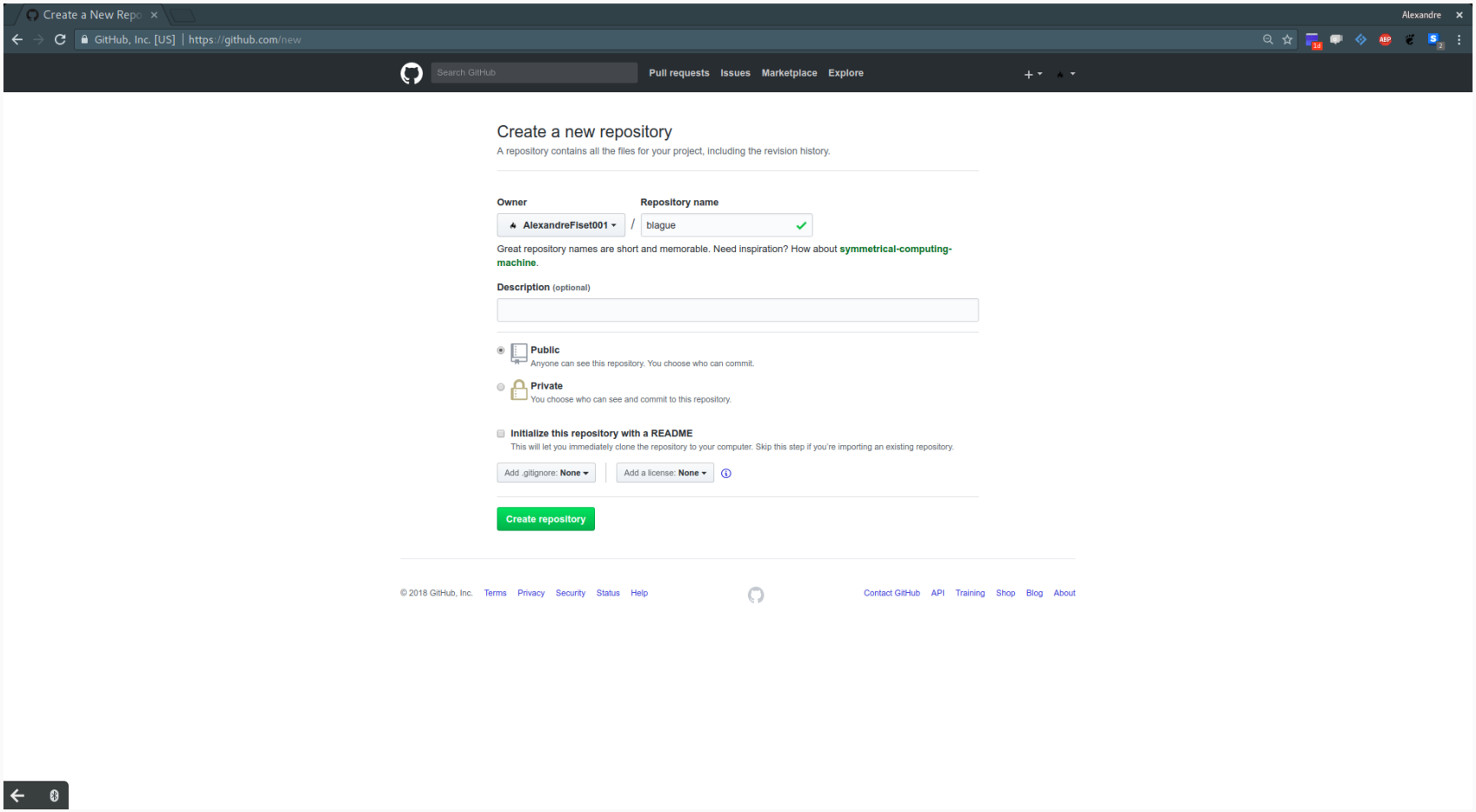
- Faire les étapes du deuxième slide de l'exercice deux.
- Enjoy :).
- N'hésitez pas a demander pour savoir ce qu'il c'est passé. #ViveLaTeX !



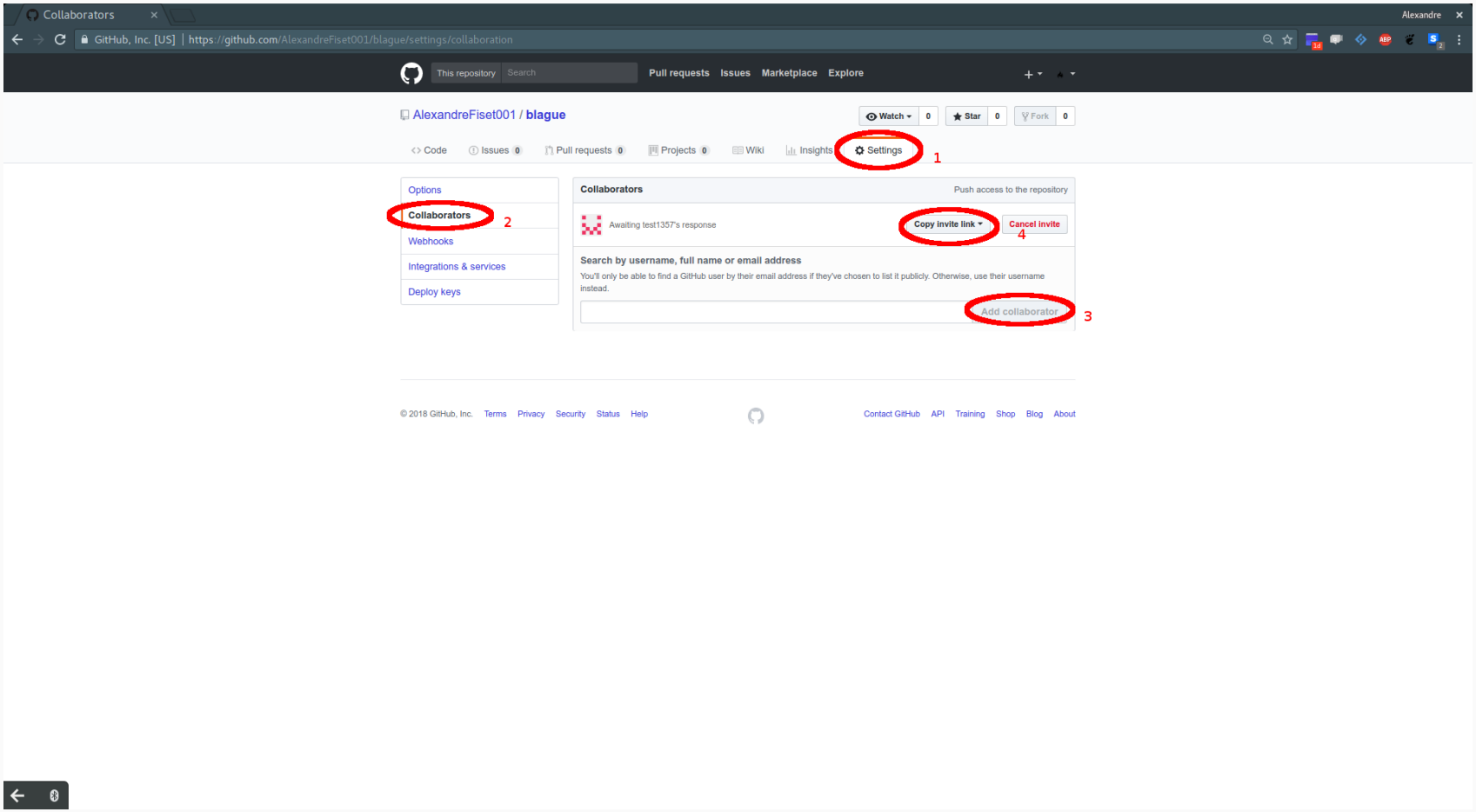
Exercice 1 : solutions



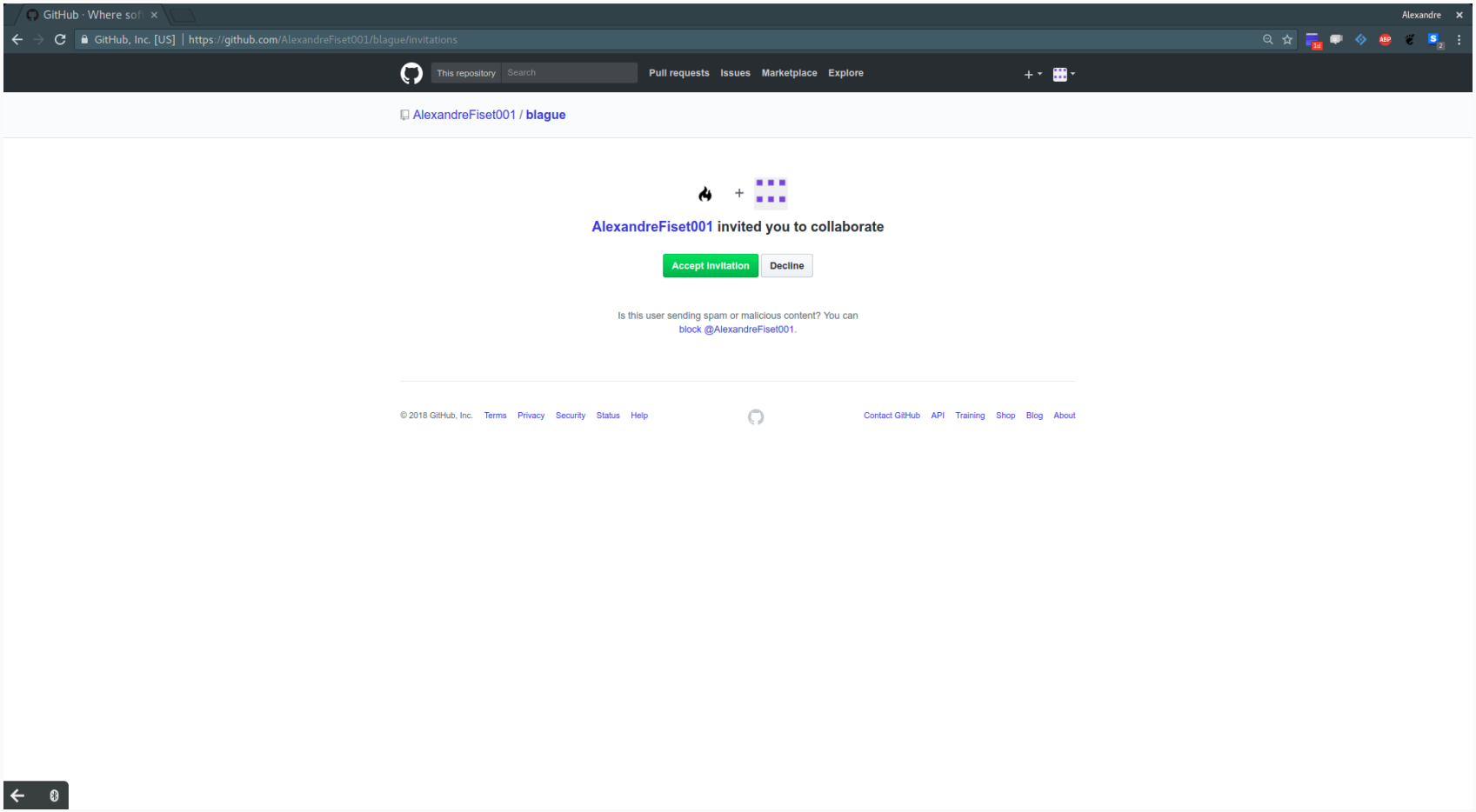
Exercice 1 : solutions



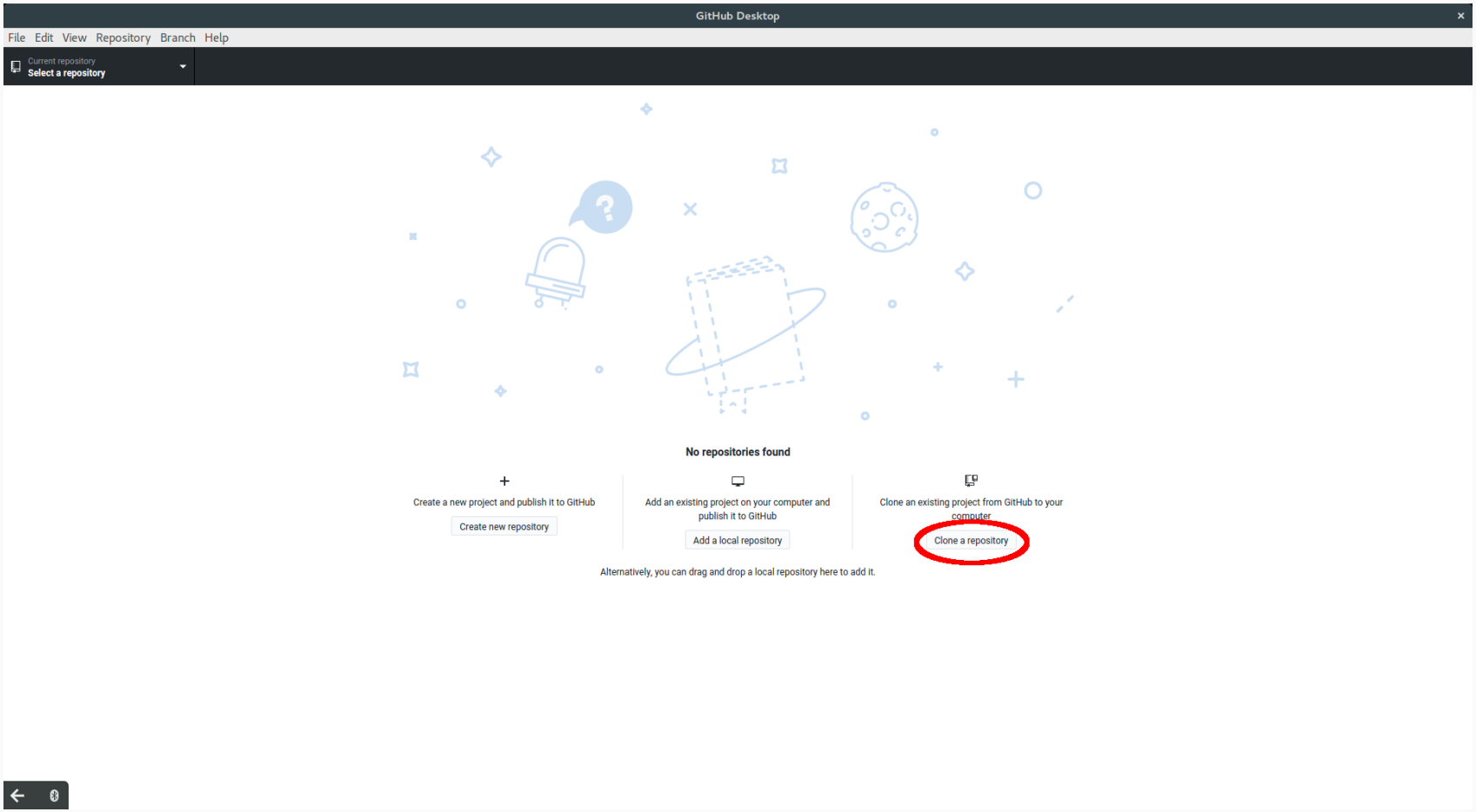
Exercice 1 : solutions



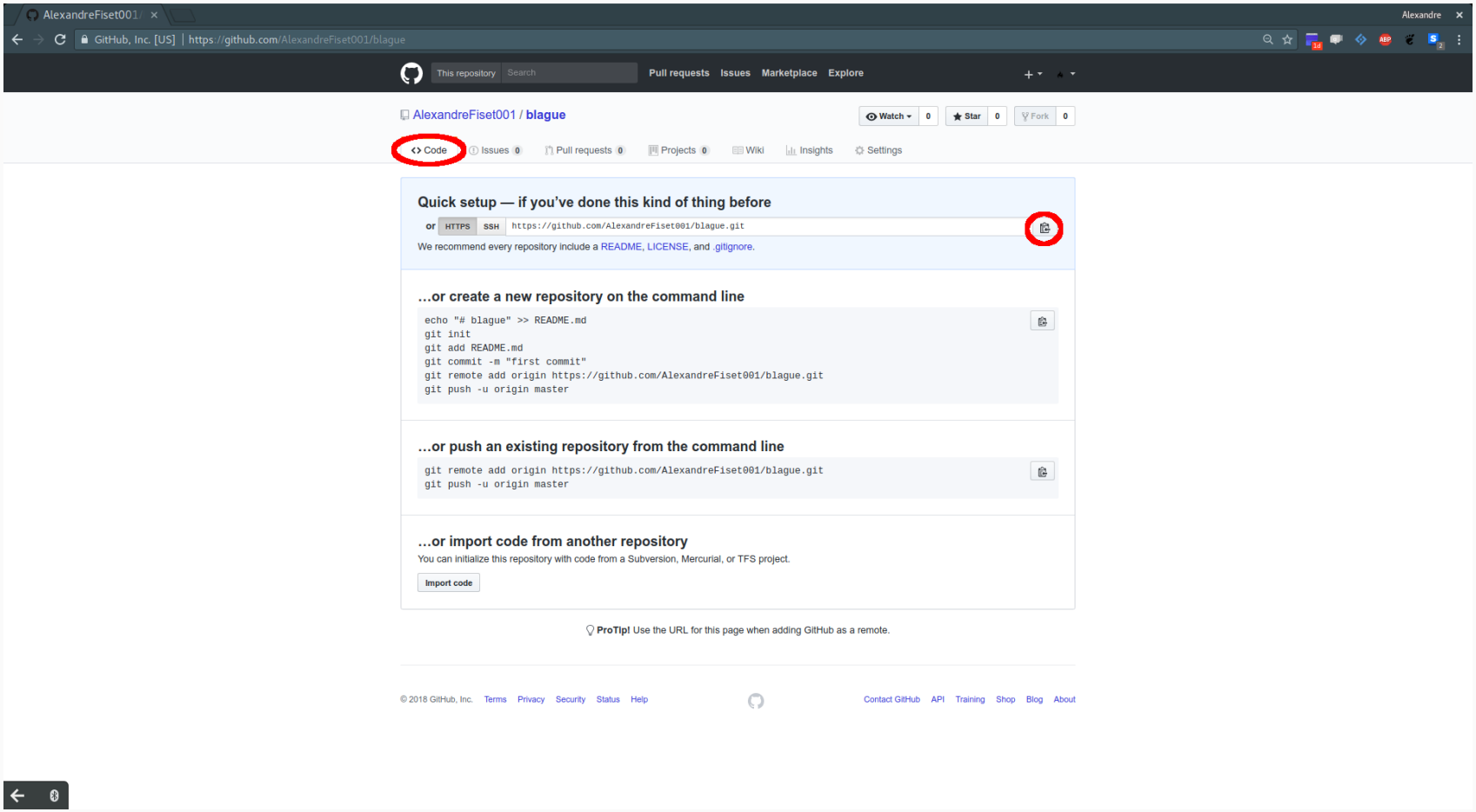
Exercice 1 : solutions



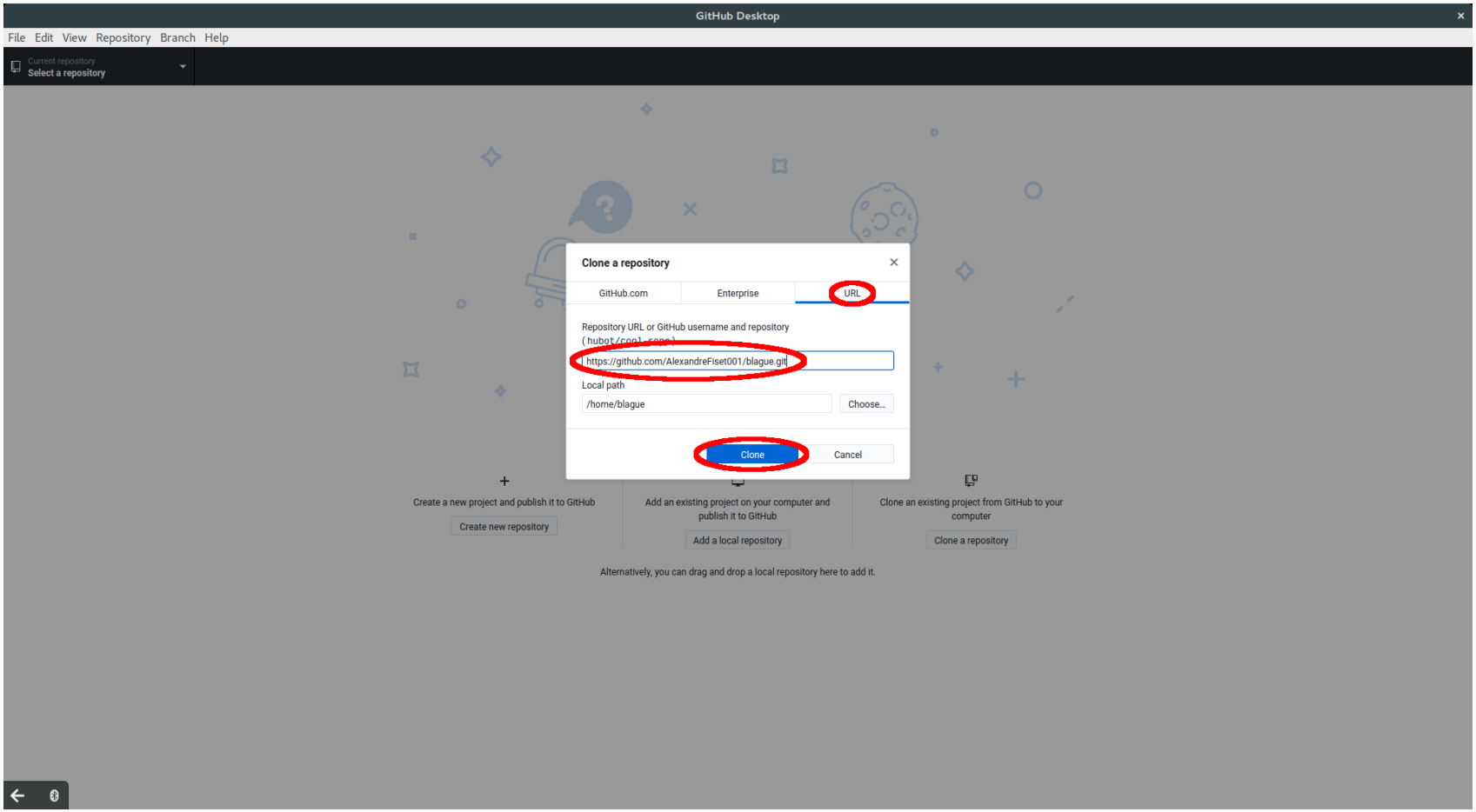
Exercice 1 : solutions



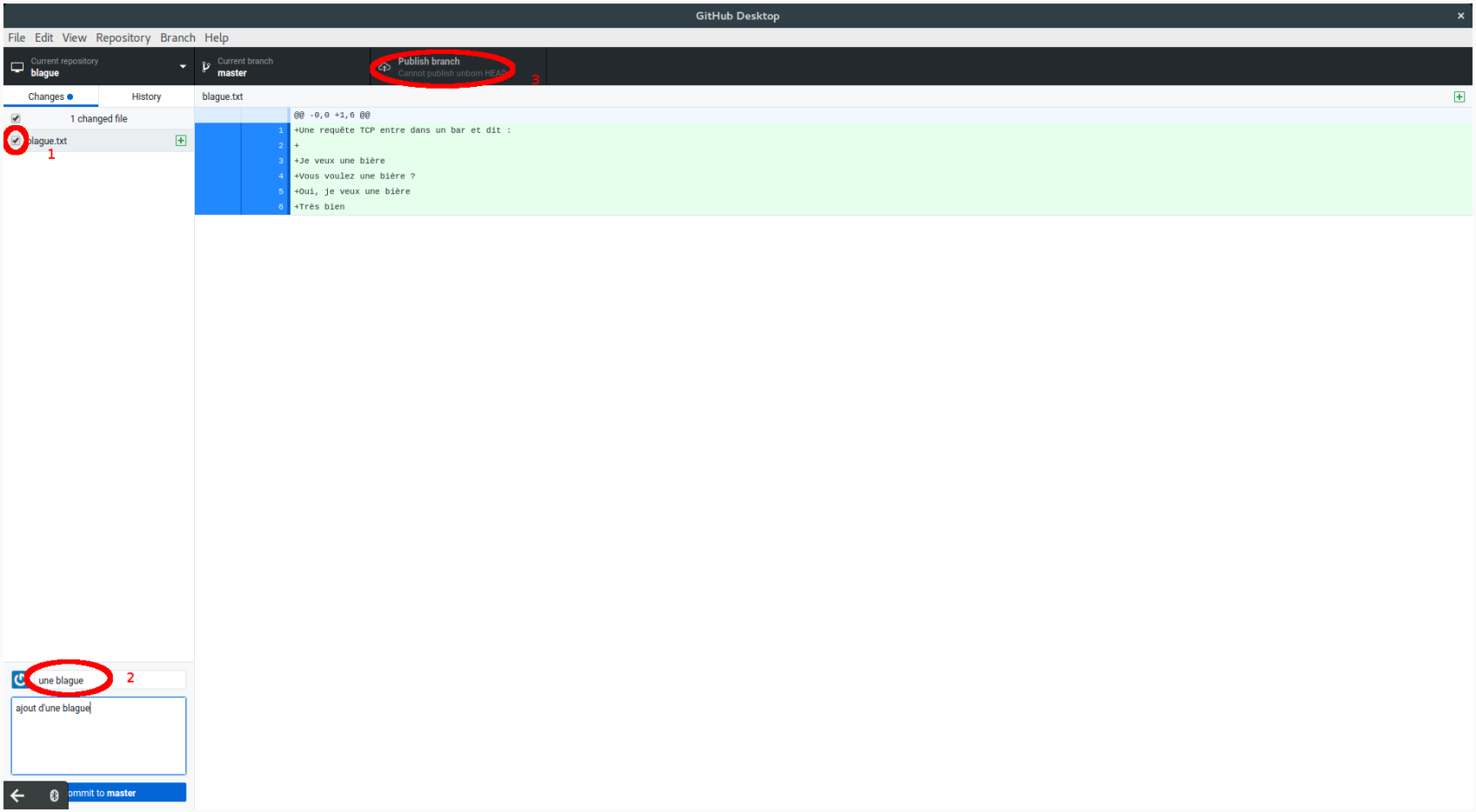
Exercice 1 : solutions



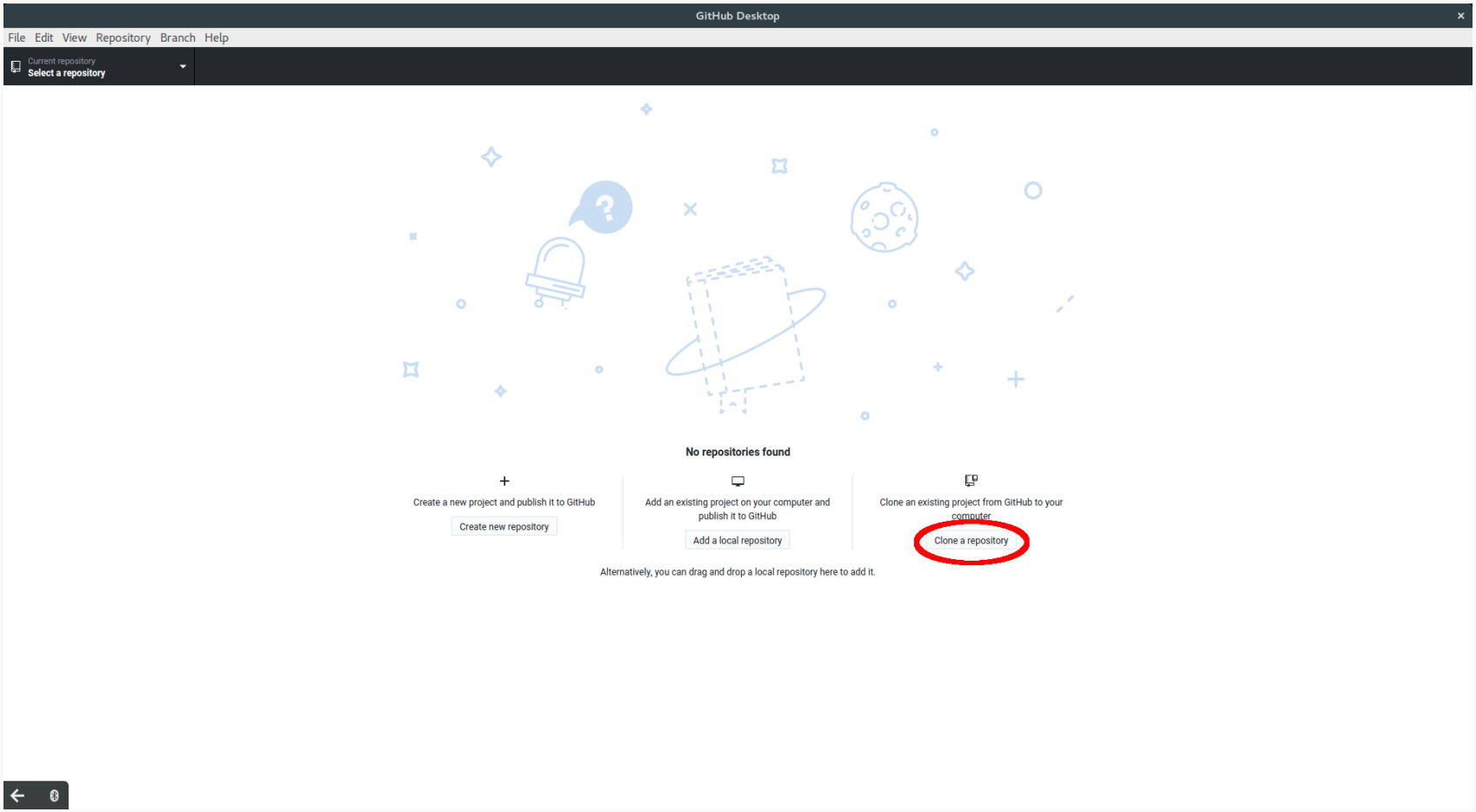
Exercice 1 : solutions



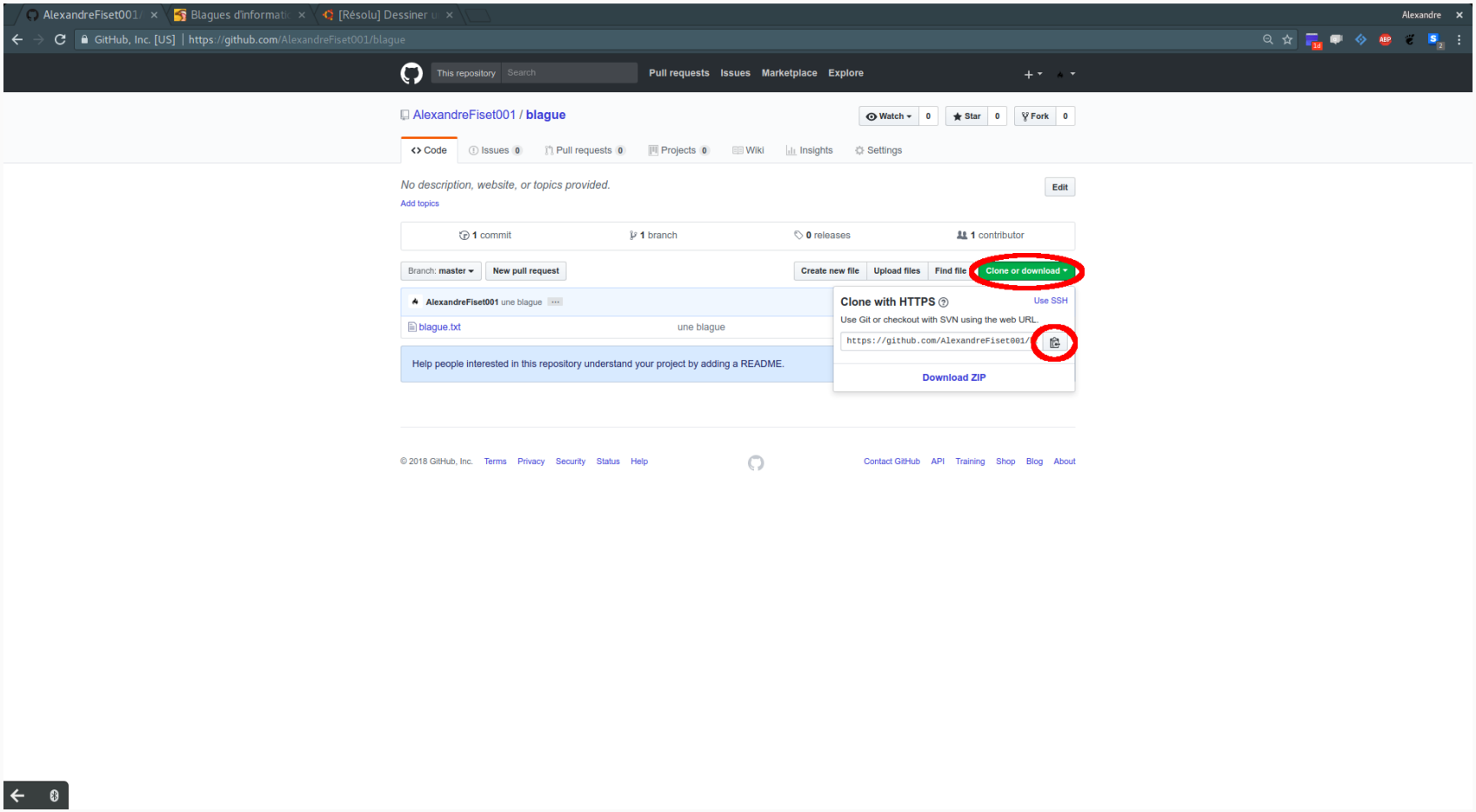
Exercice 1 : solutions



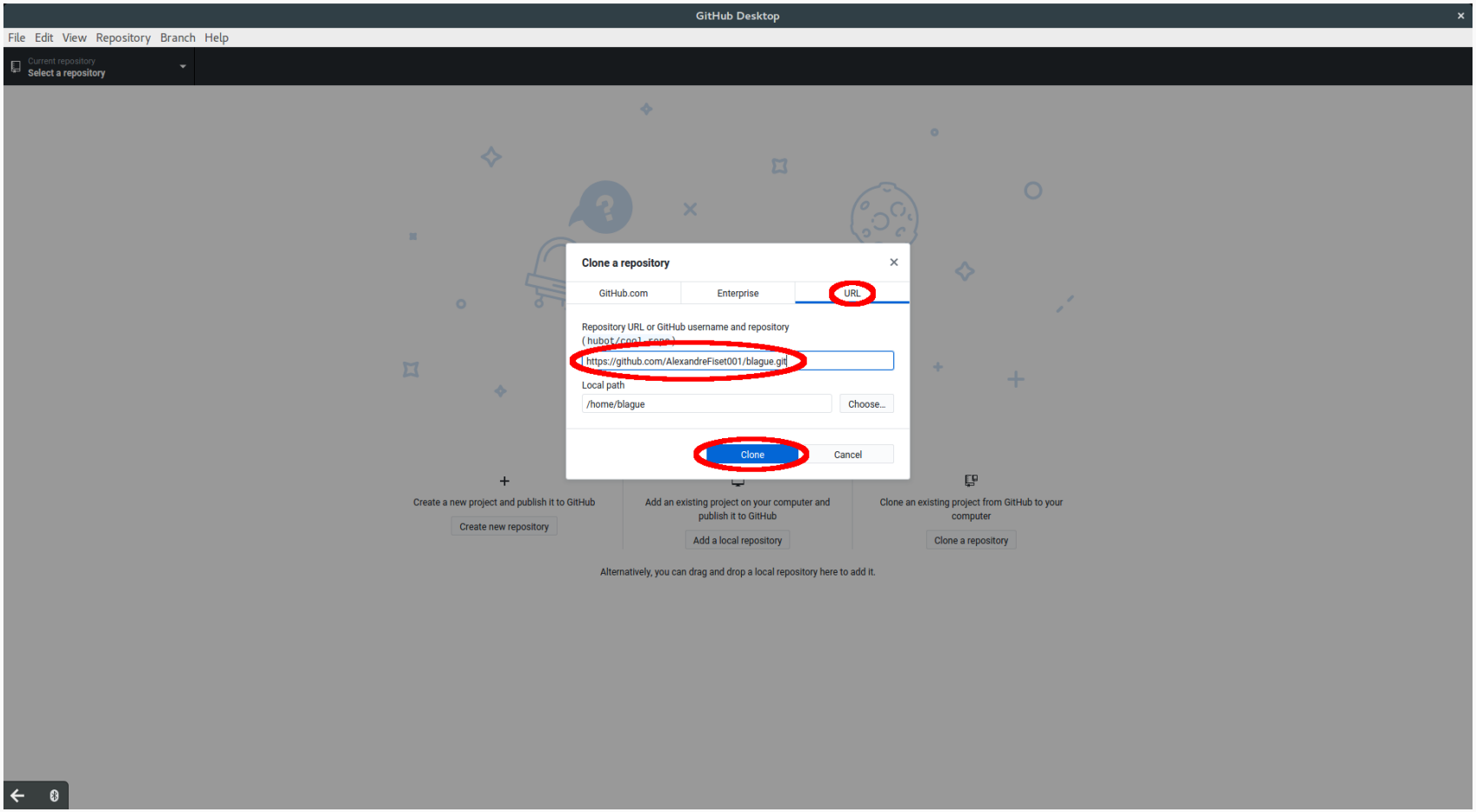
Exercice 2 : solutions



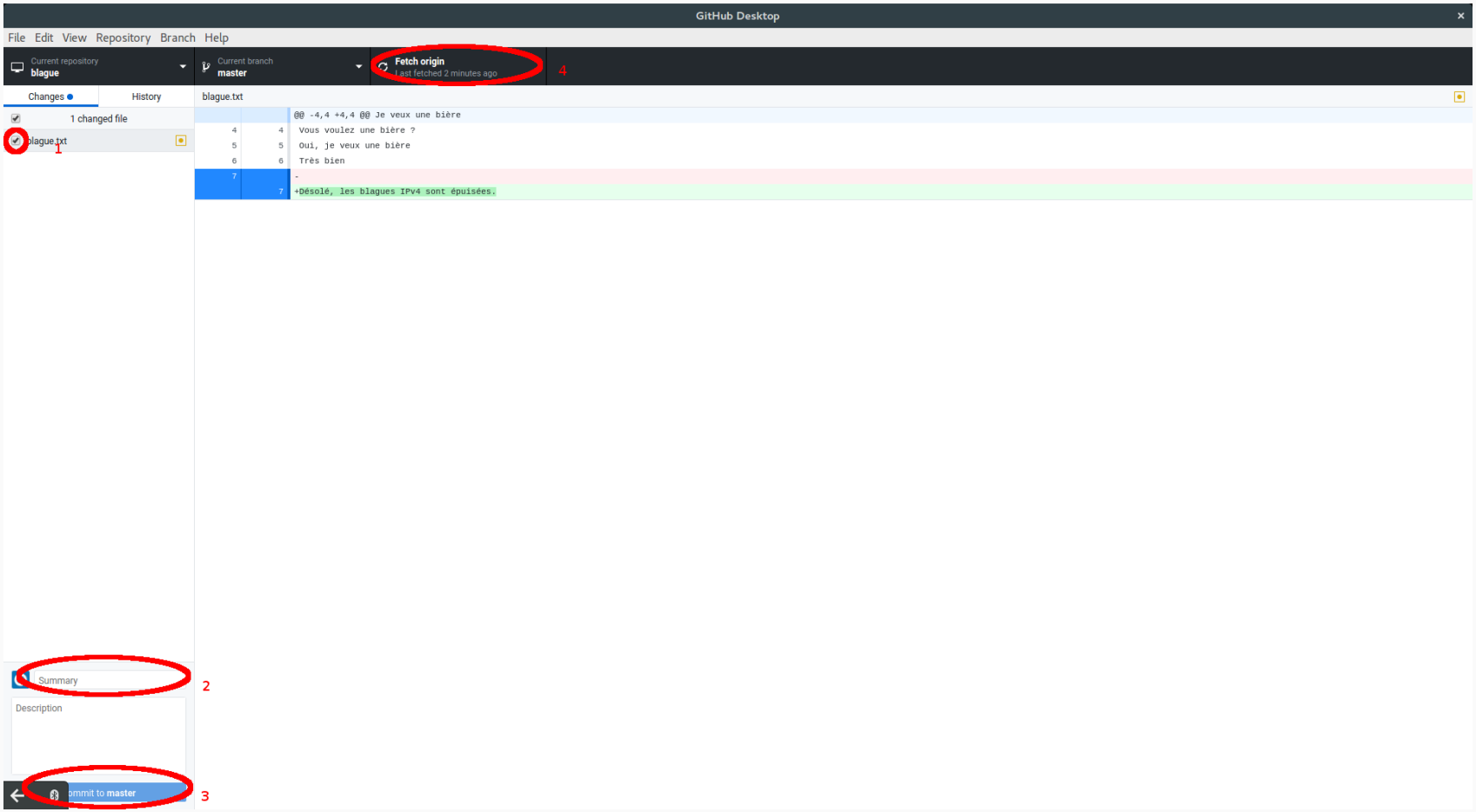
Exercice 2 : solutions



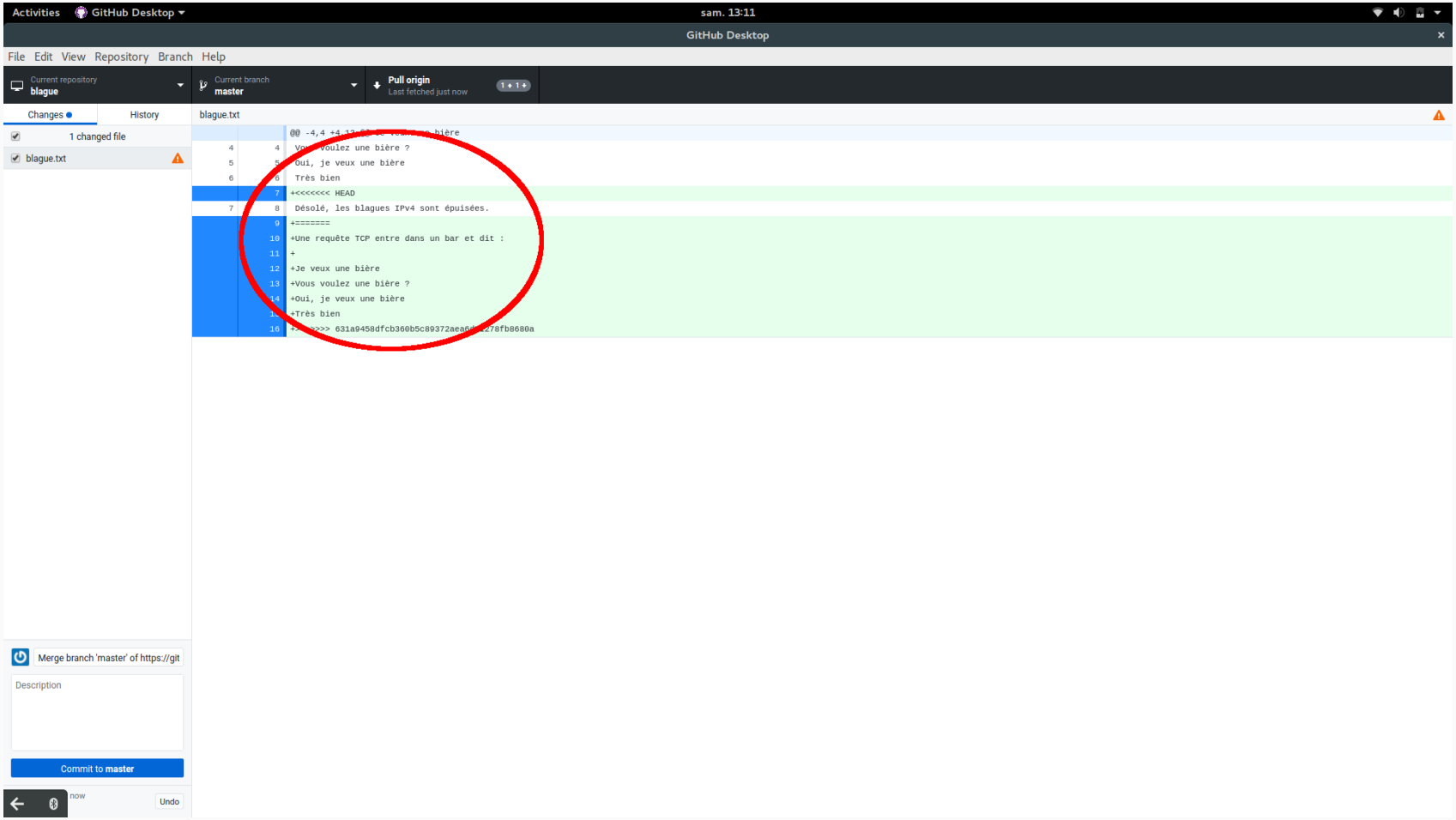
Exercice 2 : solutions



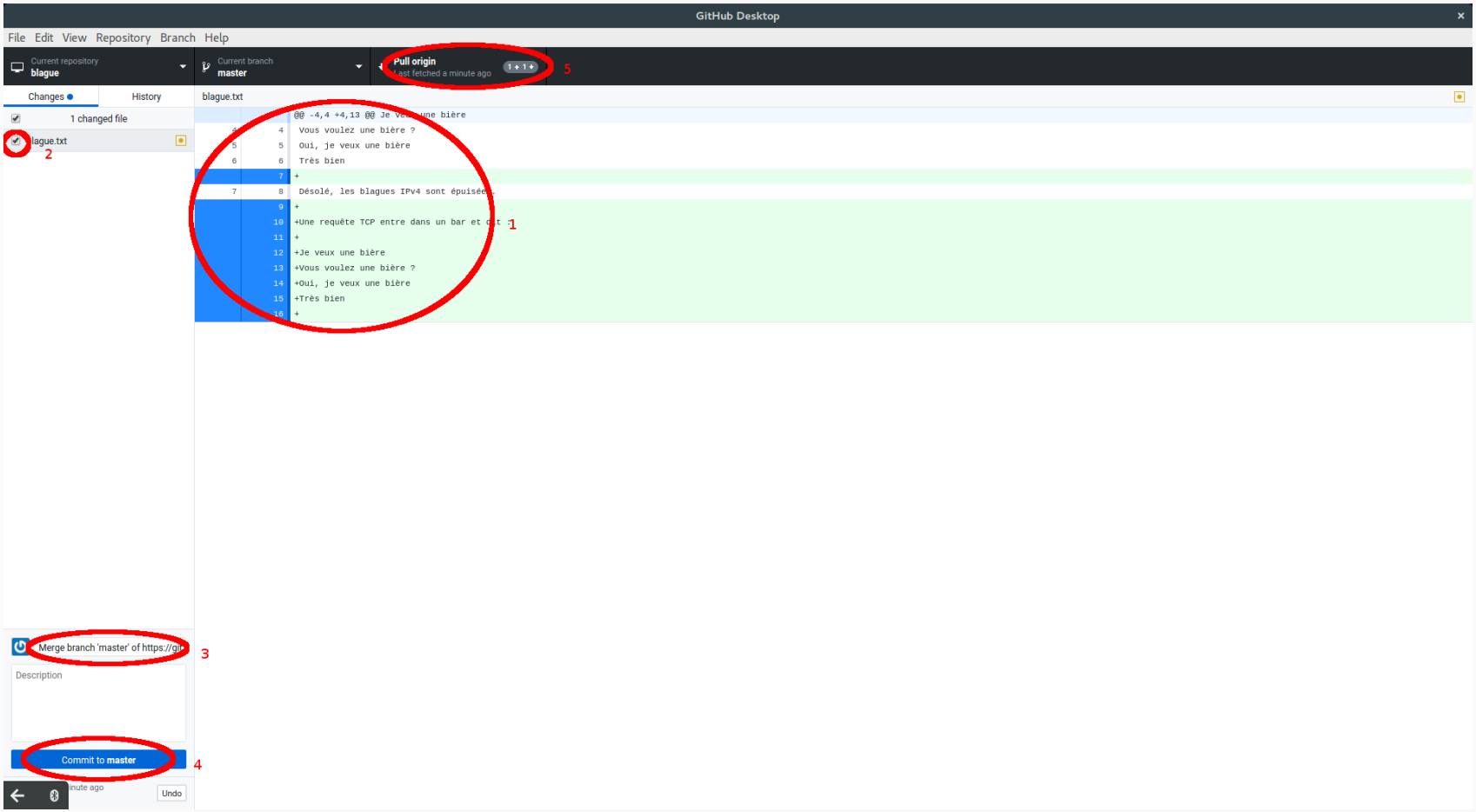
Exercice 2 : solutions



Exercice 2 : solutions



Exercice 2 : solutions



Exercice 2 : solutions

Après le merge, l'historique observé dans GitHub Desktop peut ne pas être le même chez les différentes personnes. Cela est dû au fait que GitHub Desktop n'affiche pas l'entièreté de l'historique, mais seulement les commits qu'il juge pertinents.

Pour voir tout l'historique, aller sur le site `github.com` ou bien utiliser une autre interface de git (ex. : commande `git log`).

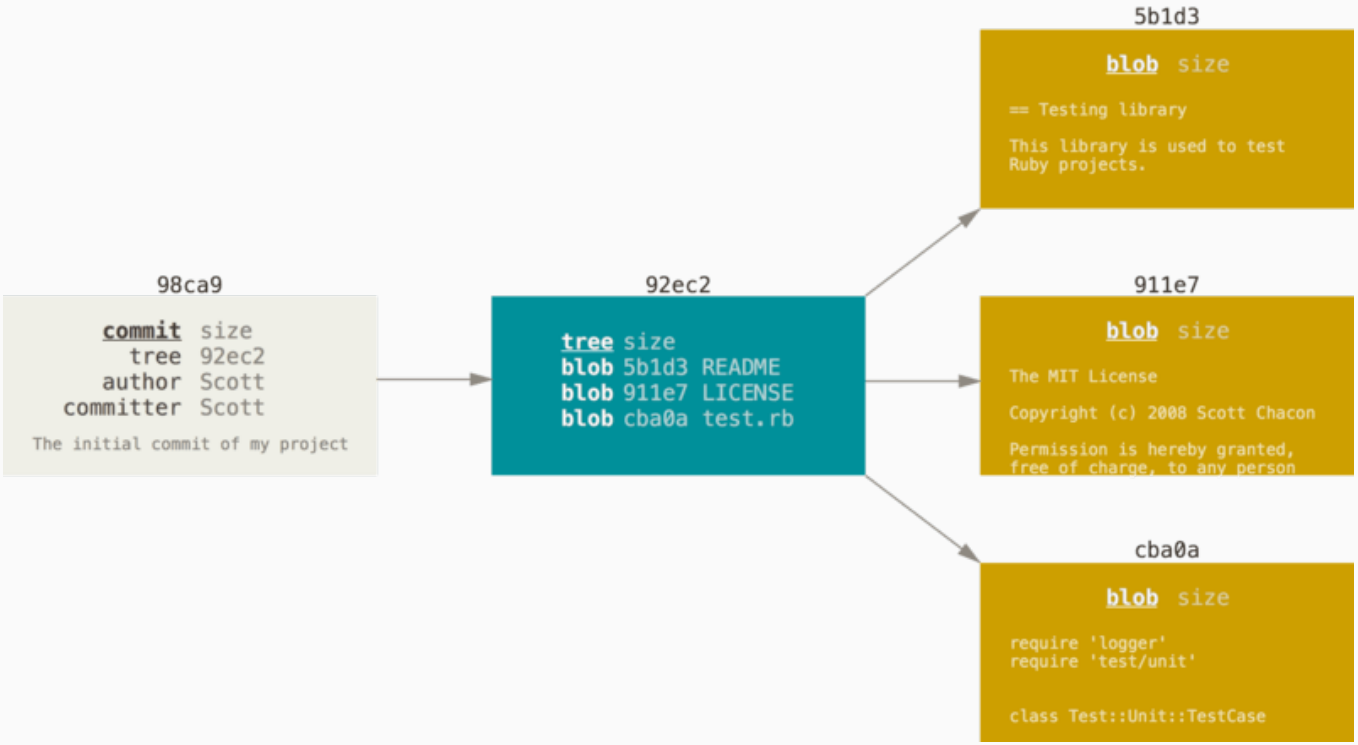
Exercice 3 : solutions

Cette solution est presque identique à la solution de l'exercice 1, sauf qu'il est impossible de faire le merge avec GitHub Desktop. Il faut utiliser une autre interface de git (voir plus loin dans les slides).

Fonctionnalités plus avancées

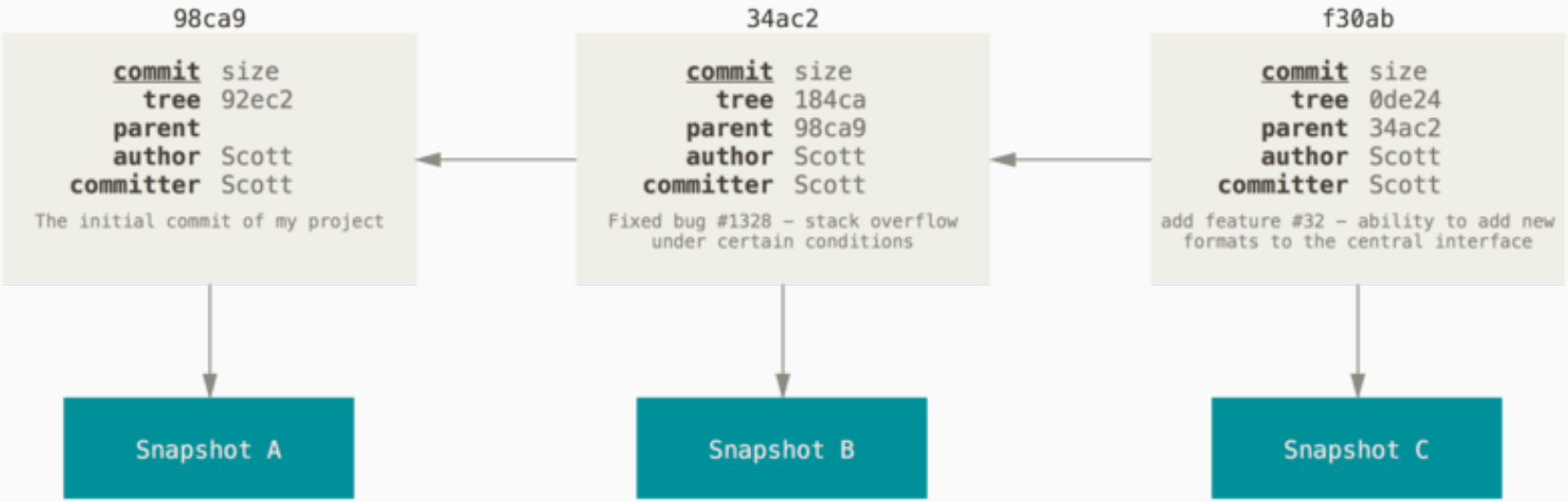
De derrière : les objets git

- Chaque commit a un identifiant : **12f87**b95caff8cbeb5ce0717528d77e27db5669c.

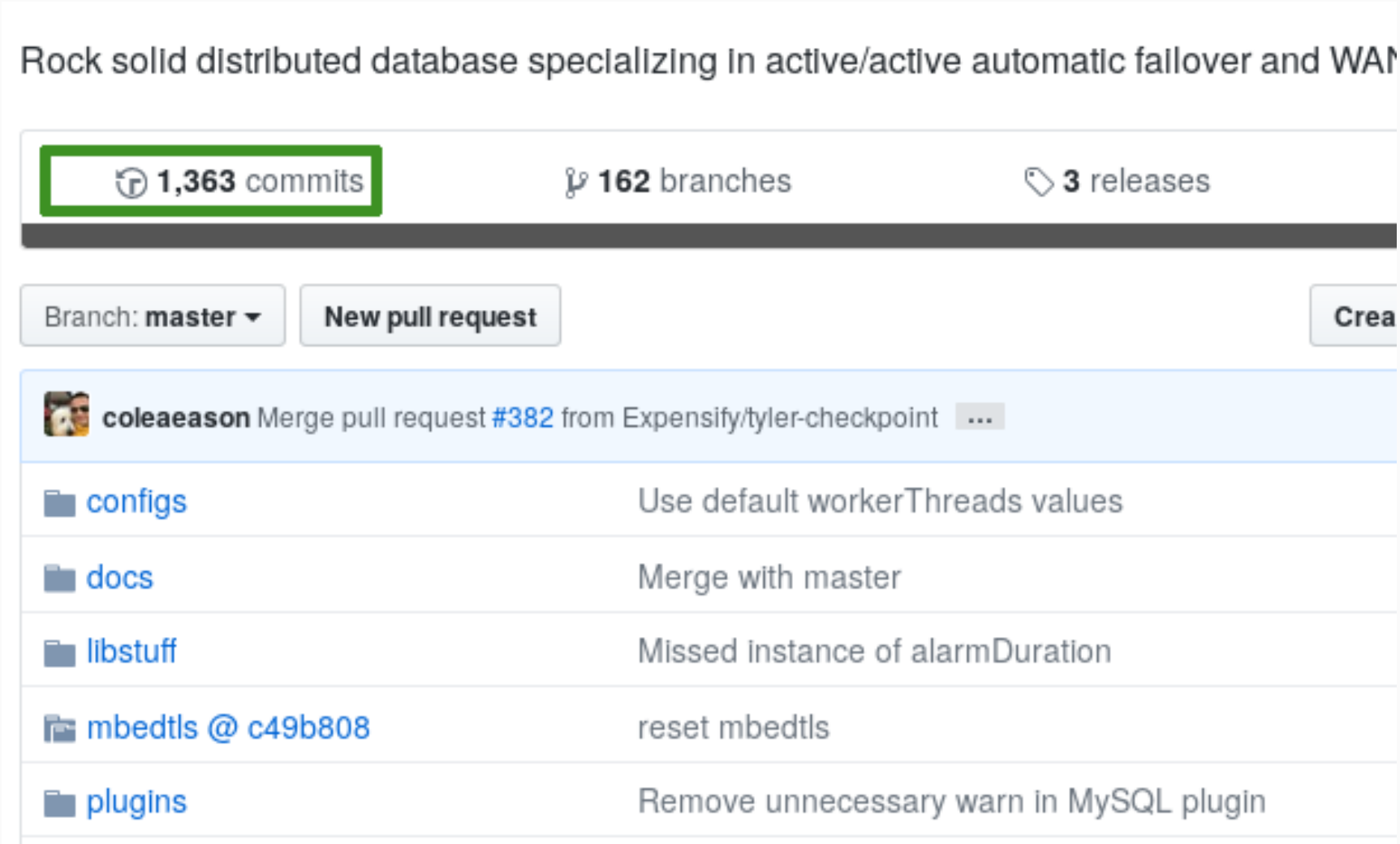


De derrière : les parents

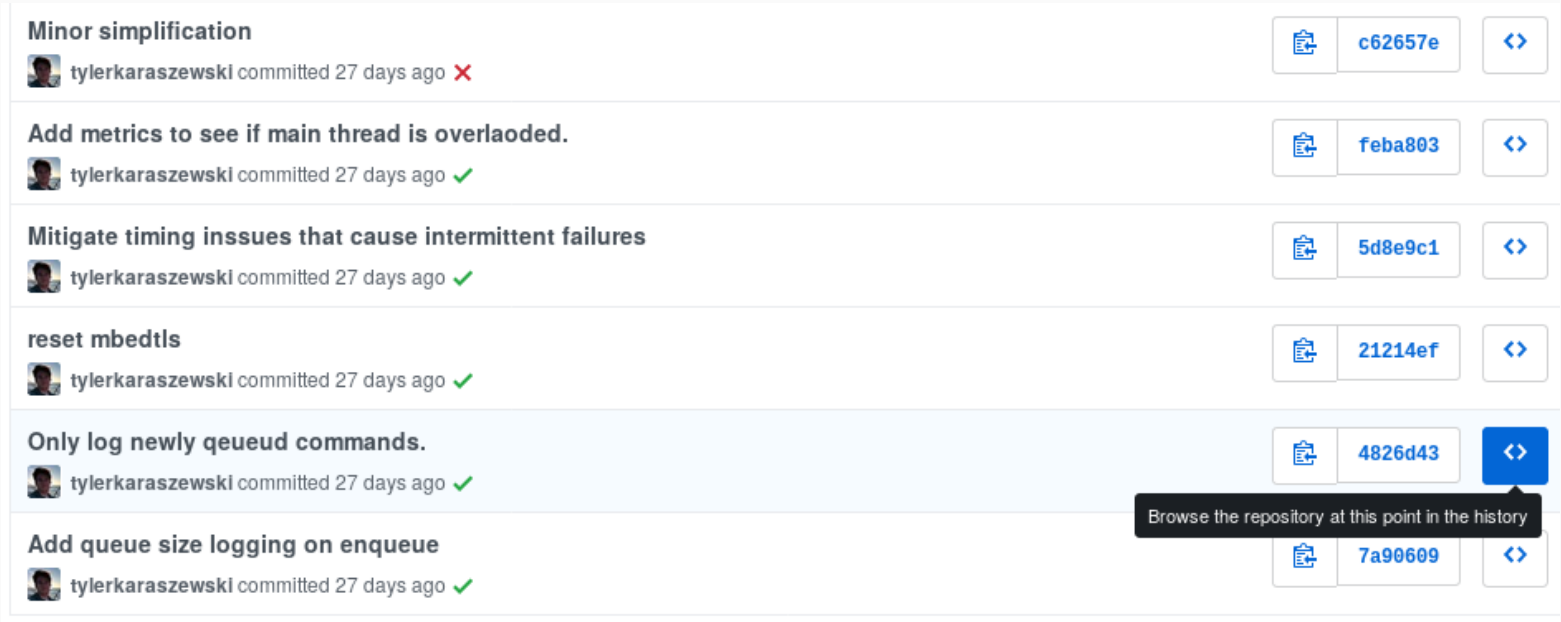
- Chaque commit a un parent.



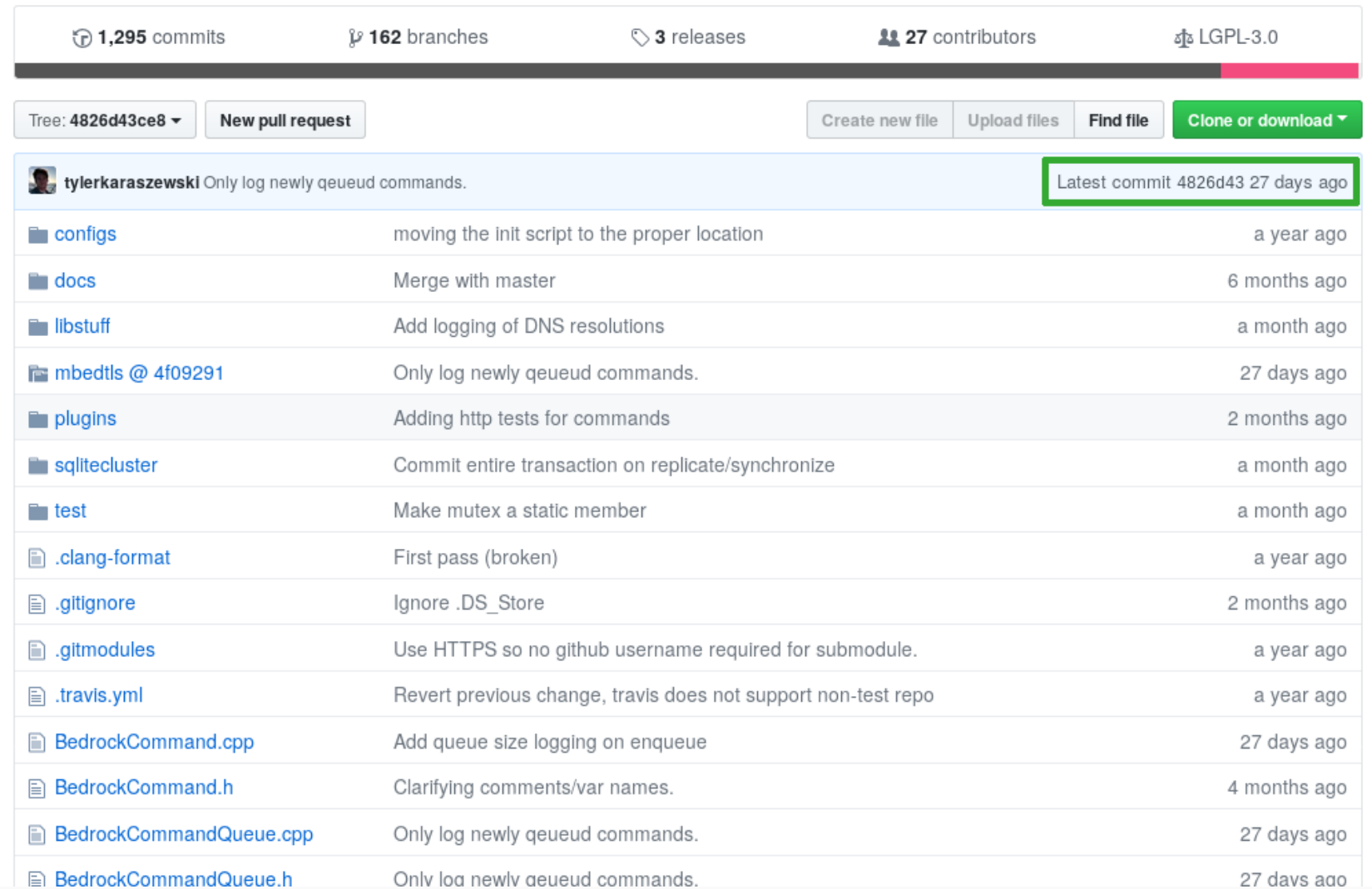
Récupérer un fichier d'un commit passé



Récupérer un fichier d’un commit passé



Récupérer un fichier d'un commit passé



1,295 commits | 162 branches | 3 releases | 27 contributors | LGPL-3.0

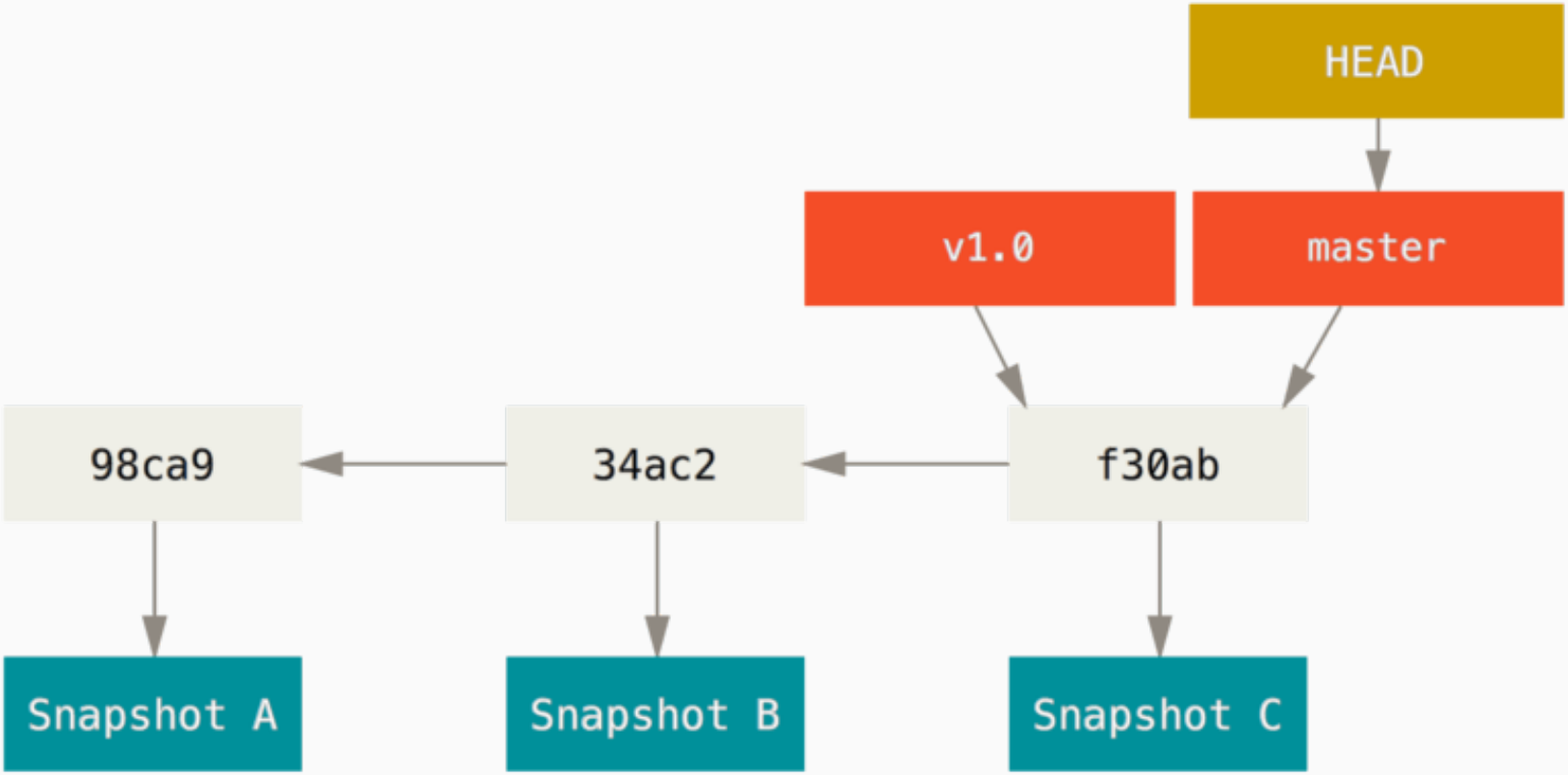
Tree: 4826d43ce8 | New pull request | Create new file | Upload files | Find file | Clone or download

tylerkaraszewski Only log newly qeueud commands. — Latest commit 4826d43 27 days ago

configs	moving the init script to the proper location	a year ago
docs	Merge with master	6 months ago
libstuff	Add logging of DNS resolutions	a month ago
mbedtls @ 4f09291	Only log newly qeueud commands.	27 days ago
plugins	Adding http tests for commands	2 months ago
sqlitecluster	Commit entire transaction on replicate/synchronize	a month ago
test	Make mutex a static member	a month ago
.clang-format	First pass (broken)	a year ago
.gitignore	Ignore .DS_Store	2 months ago
.gitmodules	Use HTTPS so no github username required for submodule.	a year ago
.travis.yml	Revert previous change, travis does not support non-test repo	a year ago
BedrockCommand.cpp	Add queue size logging on enqueue	27 days ago
BedrockCommand.h	Clarifying comments/var names.	4 months ago
BedrockCommandQueue.cpp	Only log newly qeueud commands.	27 days ago
BedrockCommandQueue.h	Only log newly qeueud commands.	27 days ago

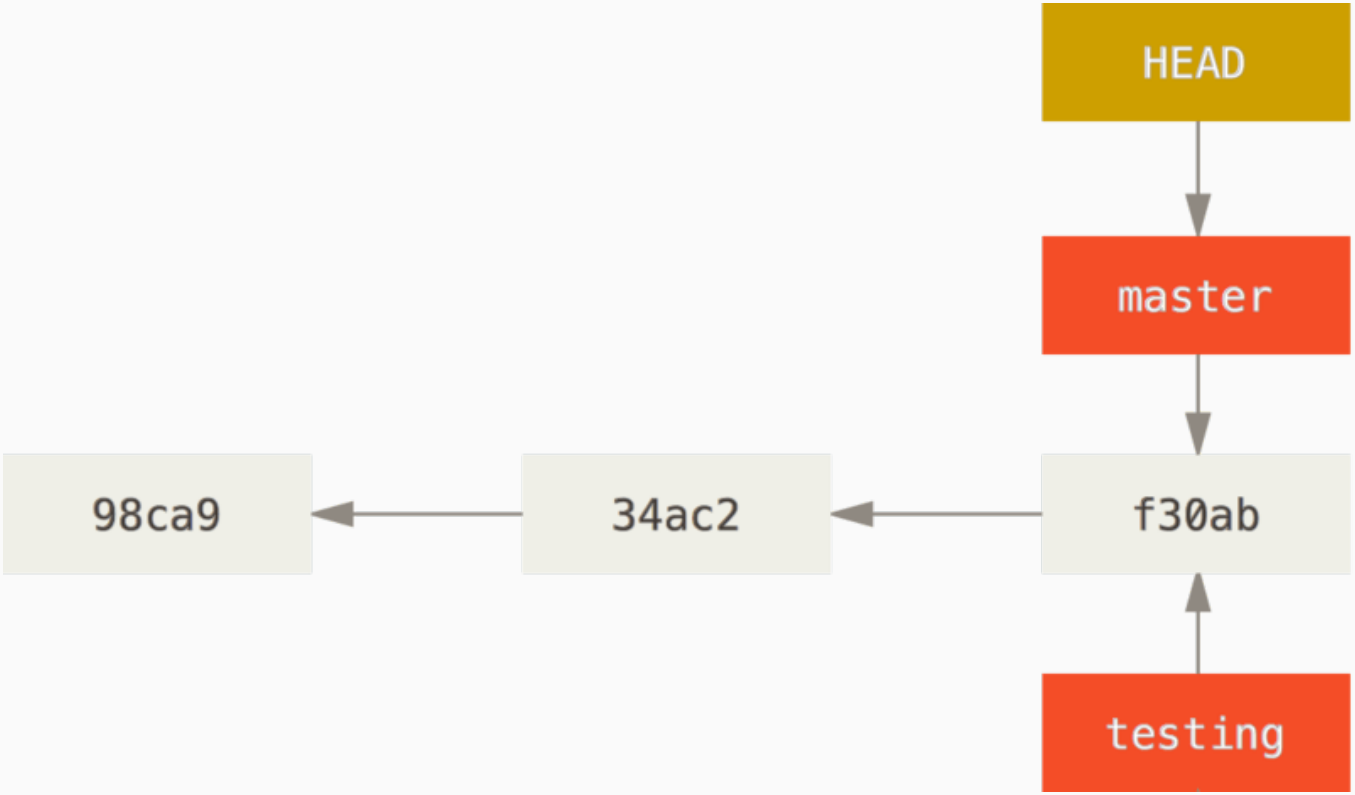
De derrière : les étiquettes

- On peut mettre des étiquettes sur des commits.
- HEAD est la position actuelle.



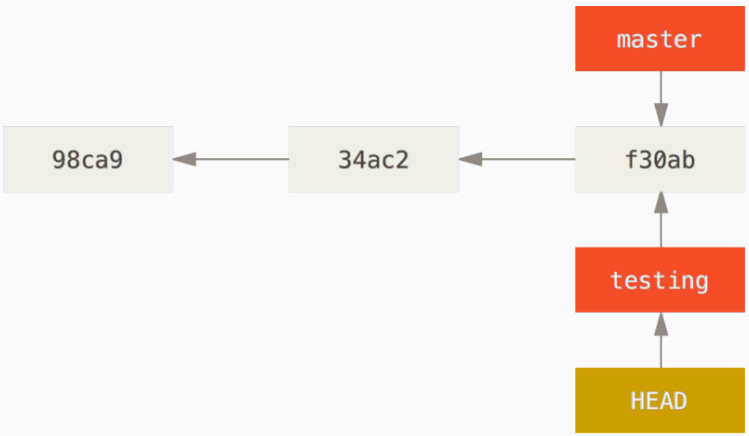
Créer une branche

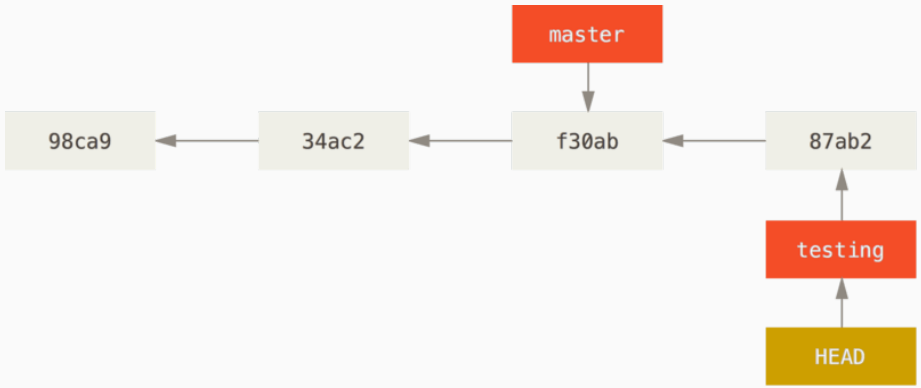
- Une branche est une nouvelle étiquette.
- La branche par défaut est `master`.



Changer de branche

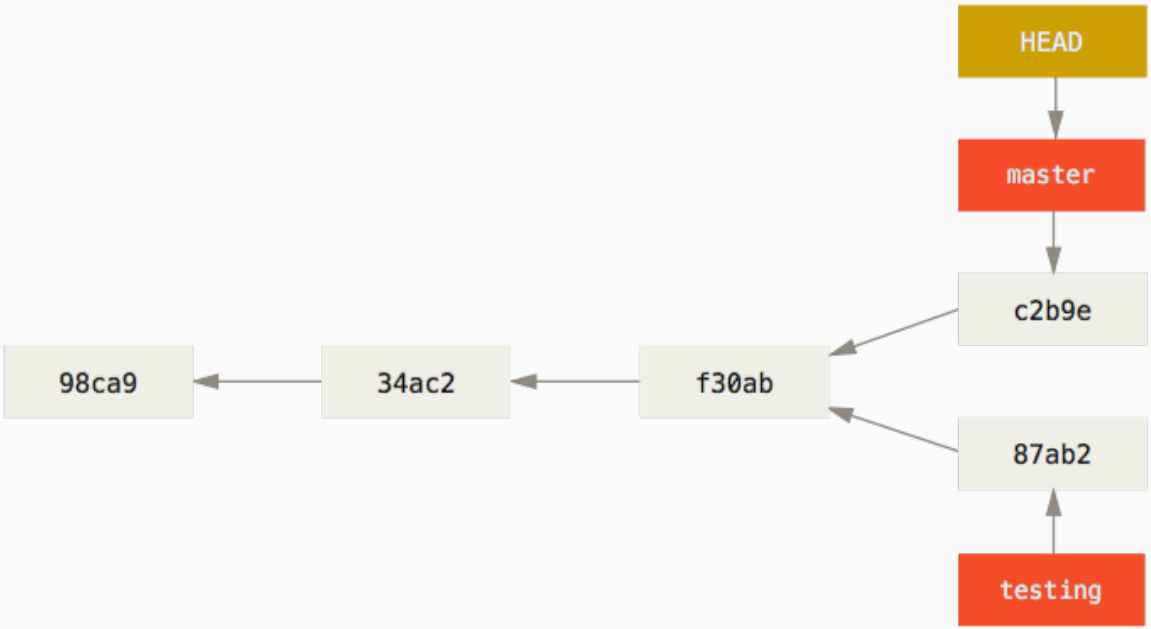
La branche courante est celle qui suit les nouveaux commits.



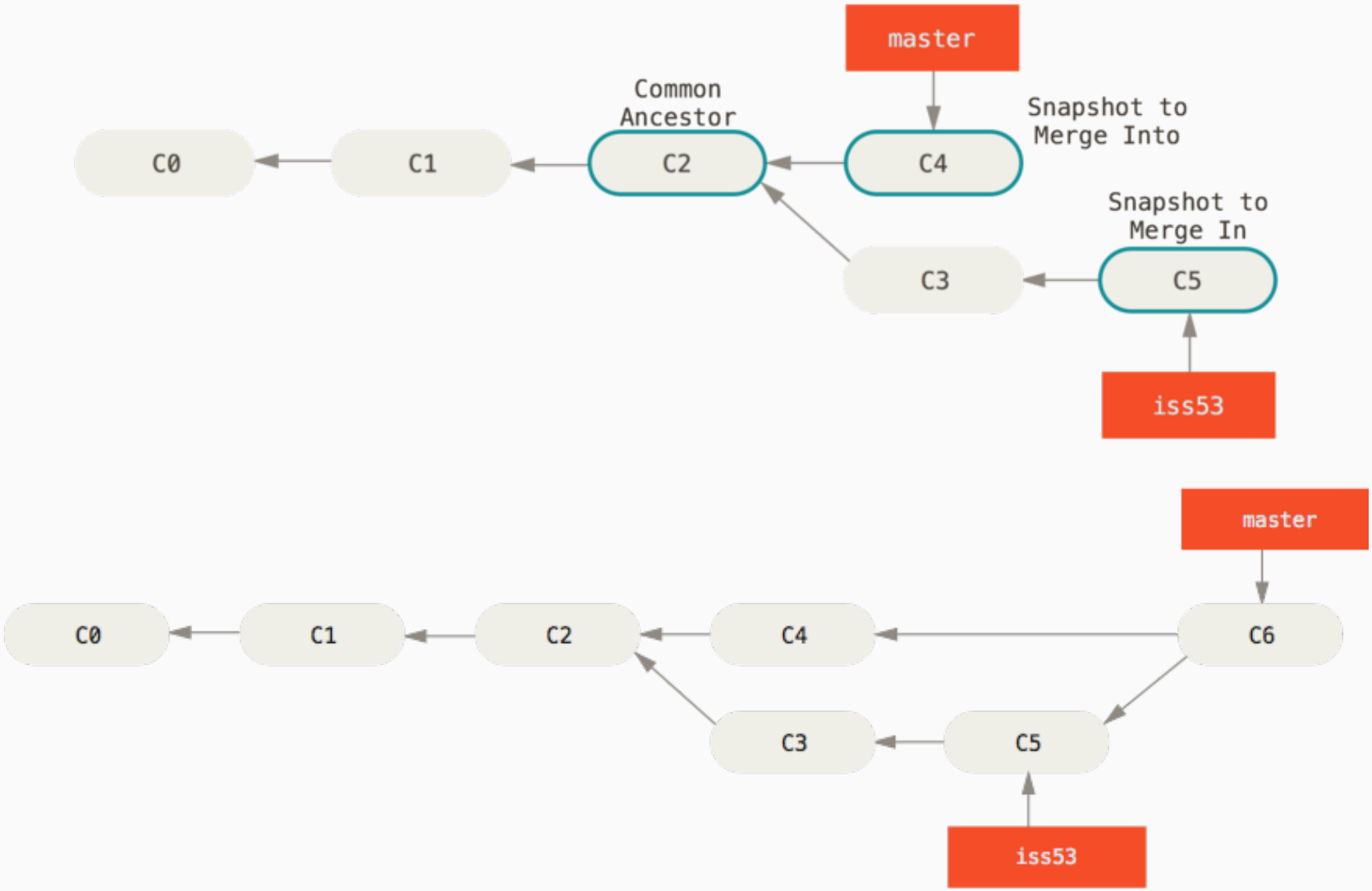


Branches divergentes

- Utilité : travailler sur des modifications indépendantes.



Fusionner des modifications

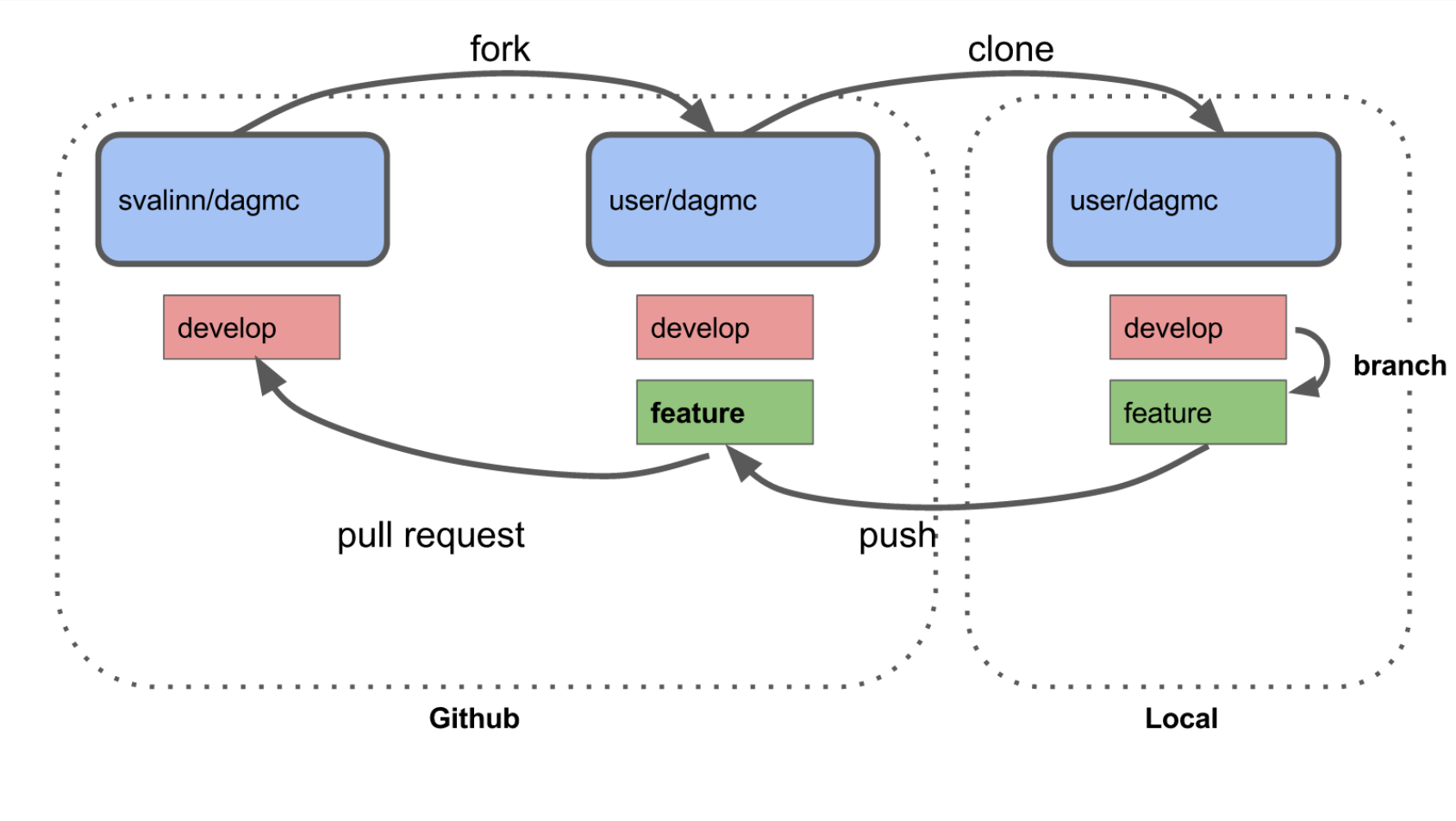


Fusionner des modifications : en pratique

Parfois il faut résoudre des conflits...

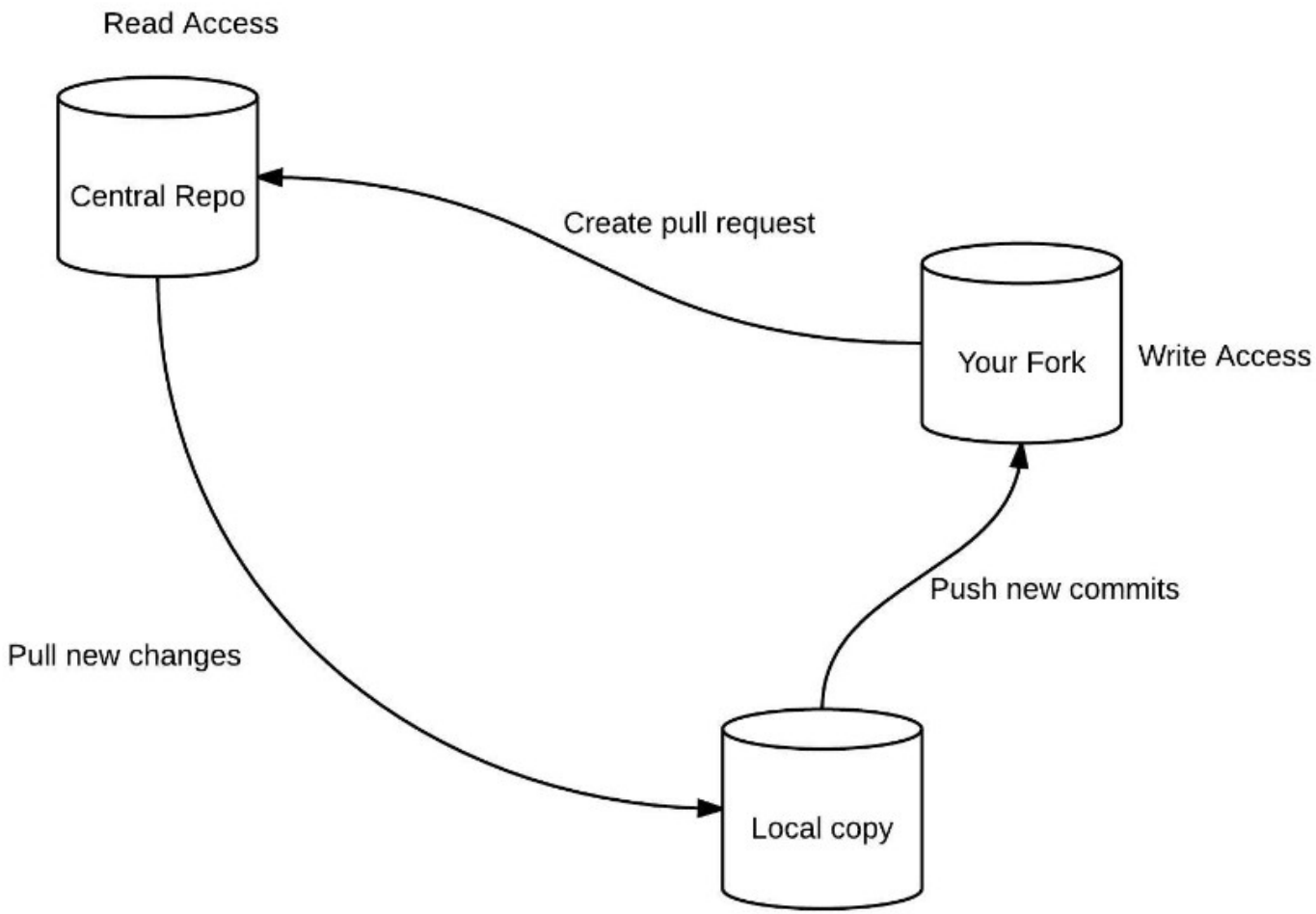
Fork – Pull Request

Une autre méthode de collaboration, très utilisée pour des larges projets et/ou projets où la contribution est ouverte à tous.

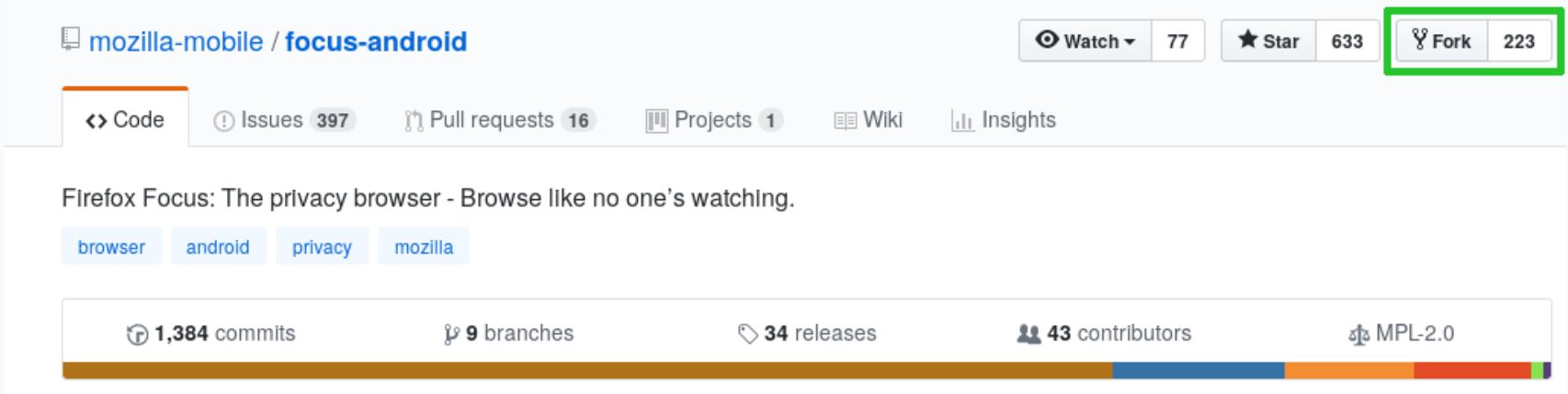


Fork – Pull Request : Méthode de travail

Voir
https://help.github.com/articles/fork-a-repo/.



Forker un dépot sur GitHub



Commandes

Déplacer la HEAD vers une branche ou un commit :

```
git checkout <ref>
```

Afficher les branches :

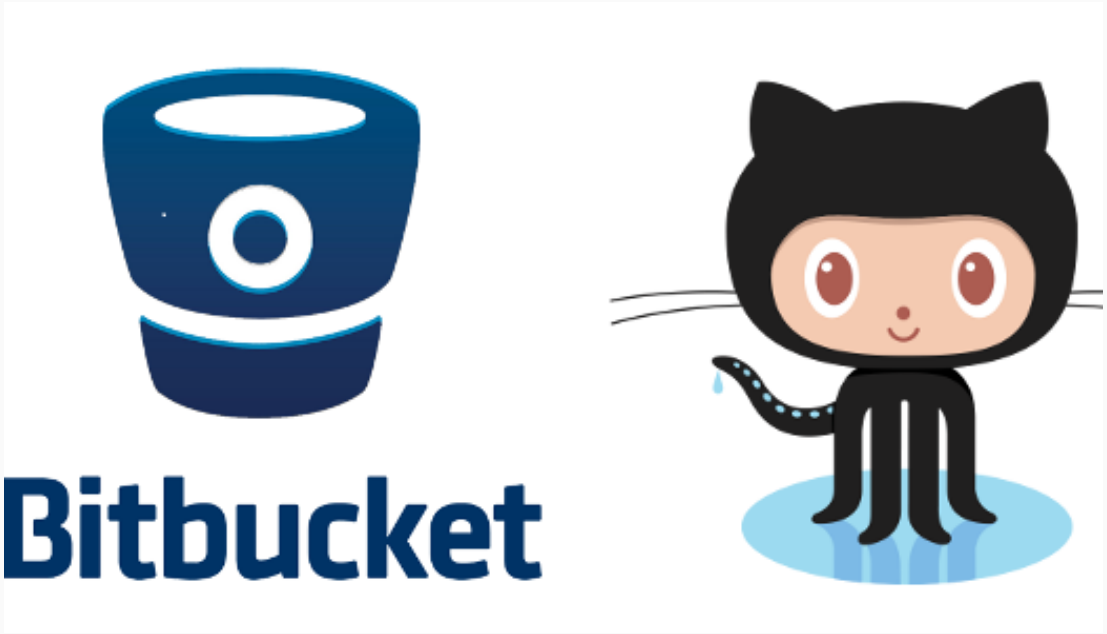
```
git branch
```

Changer l'url dun remote :

```
git remote set-url <remote> <remote-url>
```

Informations et ressources

Github, Bitbucket, Gitlab



Pratiquement identiques (tous fonctionnent avec GitHub Desktop).

Github Student Pack

Dépôts privés gratuits (tout comme sur Gitlab & Bitbucket), et d'autres avantages pour les informaticiens : `https://education.github.com/pack`.

Nécessite d'ajouter l'adresse `...@student.uclouvain.be` au compte GitHub.

Interface en ligne de commande

Utilisée par beaucoup de gens, très puissante si vous êtes à l'aise avec un terminal.

Installation :

- Ubuntu : `sudo apt-get install git`
- OS X : `https://sourceforge.net/projects/git-osx-installer/`
- Windows : `https://git-for-windows.github.io/` (déjà installé à l'UCL)

Documentation :

- La référence : Git book : `https://git-scm.com/book` : abordable, bien expliqué et très complet !
- `git help`, `git <command> help`

Autres interfaces graphiques

- `https://git-scm.com/docs/gitk` (Installé par défaut sur PC UCL)
- `https://www.gitkraken.com/`
- D'autres : `https://git-scm.com/downloads/guis`

Questions?