Lyyn

Zeste de savoir

[Git] Créer un nouveau dépôt à partir d'un dossier d'un autre dépôt

12 août 2019

Table des matières

Il y a quelques semaines, j'ai créé un dépôt sur GitHub où je place de nombreux petits bouts de codes, triés par dossiers, qui ne me servent qu'à effectuer des tests divers et variés. Cependant, un de ces essais s'est finalement transformé en véritable projet et occupe à lui seul la plupart des commits effectués sur ce dépôt. Voici donc un problème : il faudrait que je crée un nouveau dépôt qui ne contiendra que ce projet ! Surtout si je souhaite le rendre open-source.

Alors évidemment, j'aurais très bien pu créer mon dépôt, un nouveau dossier local et faire un simple copier/coller des fichiers.. Mais quid de **l'historique des commits** ? C'est tout de même important, surtout à mes yeux ! Et pour bien faire, je souhaite également **changer d'hébergeur** : passer de GitHub à BitBucket. Ah, dernière chose à ne pas oublier : les premiers commits ont mélangé divers fichiers des multiples projets... Et je ne veux pas en garder une seule trace !

Résumons donc mes exigeances :

— Créer un dépôt sur BitBucket
— Déplacer mon projet du dépôt "fourre-tout" jusqu'à ce nouveau dépôt
— Ne subir aucune perte de fichier
— Ne pas perdre l'historique des commits
— Avoir des commits filtrés (qui ne concernent que mon projet)

Sachant que **Git** est un outil vraiment puissant, simple à prendre en main et très fiable, je me suis dit qu'il y avait une chance de parvenir à atteindre au moins la moitié des objectifs... Au final, j'ai réussi à **tous les accomplir** ! *Et vous ne devinerez jamais comment j'y suis arrivé !*

1. Pré-requis

FIGURE 1. – Un chat-pieuvre, tout un symbole...

Il faut posséder **git** sous forme de terminal. Sur mon ordi, j'ai choppé **Git Shell** en installant **GitHub for Windows** à cette adresse : https ://windows.github.com

Un minimum de connaissance de **git** sera nécessaire. Je n'explique pas la création d'un dépôt, le clonage, le pull, le commit, ... Ce sont des notions primaires, très facile à aborder. C'est pourquoi je n'en parlerai pas.

Il faudra également posséder des doigts agiles et une certaine concentration : nous sommes en lignes de commande, une faute de frappe peut tout changer !

Pour finir, il faudra créer un dépôt en ligne à l'avance, sur http ://www.sourcetreeapp.com/ ou https ://bitbucket.org/ . D'autres sites utilisant **Git** fonctionneront aussi, il faudra juste adapter la ligne adéquate (ce sera indiqué à l'étape nécessaire).

... Et c'est tout. Je conseillerais aussi **SourceTree** d'Atlassian (ceux qui ont fait BitBucket !) qui est un logiciel permettant de gérer des dépôts **Git** ou **Mercurial**, de façon gratuite. Il est vraiment agréable à utiliser et permet à un néophyte d'utiliser le versioning sans trop se prendre la tête (parfait pour moi, ahah). Vous pouvez le télécharger ici : http ://www.sourcetreeapp.com/

2. Explications

Il ne faudra pas beaucoup d'étapes pour accomplir ce que l'on recherche ici. En premier, on va d'abord cloner le premier dépôt dans un nouveau dossier, dossier qui sera la cible du nouveau dépôt.

Ensuite, on filtrera le contenu du nouveau dépôt pour ne garder que le dossier qui nous intéresse. Le contenu de ce dernier sera mis à la racine du dossier du dépôt. Il ne restera plus qu'à filtrer aussi les commits, passer un petit coup de balayette et balancer le tout sur notre nouveau dépôt.

Facile, donc.

3. Déplacer le dossier vers un nouveau dépôt

… Tout en gardant l'historique des commits !

Pour bien se situer et illustrer facilement, je considère qu'on se trouve un niveau au-dessus des dossiers de tous les dépôts :

```
<dossier Git>
|
| -- <dépôt starwars>
|    -- ...
|
| -- <dépôt fourre_tout>
|    -- <banane>
|    -- <pac_man>
|    -- <coffee>
|
```

Je souhaiterais déplacer le sous-dossier banane vers un nouveau dossier au même niveau que le dossier fourre_tout afin de le cibler comme nouveau dépôt.

3.1. 1) On clone le dépôt !

D'abord, on clone l'entièreté du dépôt actuel dans un nouveau dossier qui servira de cible au nouveau dépôt :

```
> git clone --no-hardlinks fourre-tout projet_banane
```

le nouveau dossier **projet_banane** sera créé au même niveau que le dossier fourre-tout, il contiendra l'entièreté des fichiers de ce dernier. Si vous avez déjà créé le dossier banane à ce niveau, il faudra s'assurer qu'il soit vide. *(Le nom importe peu, j'aurais pu lui donner le nom banane mais pour éviter les confusions, je l'ai appelé ainsi)*

On se retrouve donc avec ceci :

```
<dossier Git>
|
| -- ...
|
| -- <dépôt fourre_tout>
|    -- <banane>
|    -- <pac_man>
```

```
|    -- <coffee>
|
| -- <dépôt projet_banane>
|    -- <banane>
|    -- <pac_man>
|    -- <coffee>
```

3.2. 2) On trie !

Ensuite on se déplace à l'intérieur du nouveau dossier **projet_banane**, on supprime tout ce qui n'est pas lié à ce projet et on met le contenu du dossier **banane** à la racine du dossier **projet_banane**. Tout ça en seulement trois lignes !

```
> cd projet_banane
> git filter-branch --subdirectory-filter banane HEAD -- --all
> git reset --hard
```

La deuxième ligne va filtrer tout (fichiers et commits) et virer tout ce qui n'est pas en rapport avec notre projet banane La troisième ligne quant à elle va remettre le dépôt sur le dernier commit du projet concerné

Au niveau des fichiers... :

```
<dossier Git>
|
| -- ...
|
| -- <dépôt fourre_tout>
|    -- <banane>
|    -- <pac_man>
|    -- <coffee>
|
| -- <dépôt projet_banane>
|    -- ... (fichiers de ./banane)
```

3.3. 3) Et on nettoie !

Maintenant, nous avons les bons fichiers au bons endroits. Mais la base de données de git contient encore toutes les références à tous les autres dossiers/fichiers/commits des autres projets que nous venons d'effacer. Nettoyons tout ça :

```
> git gc --aggressive
> git prune
```

Et *voilà* ! Nous avons fini de déplacer proprement les fichiers et les commits, il ne nous reste plus qu'à...

4. Actualiser le dépôt

FIGURE 4. – On dirait un cyclope droïde. Ou un seau futuriste !

Ici, c'est la partie la plus simple. Le changement du dépôt d'origine se fait en deux étapes, et il est nécessaire de le faire afin que notre dossier soit bien lié au nouveau dépôt et non plus à l'ancien.

1) En premier, on supprime le lien vers le dépôt actuel (qui est donc celui du premier projet, ce qui ferait des conflits évidemment)

```
> git remote rm origin
```

2) Ensuite, on indique la nouvelle adresse de référence. Dans mon cas, c'est un dépôt chez **BitBucket**, mais je vous mets également la version pour **GitHub**. Pour les autres sites, vous verrez vite ce qu'il faut modifier

```
(GitHub)
> git remote add origin
    https://github.com/<nom_d'utilisateur>/<nom_du_dépôt>.git

(BitBucket)
> git remote add origin
    https://<compte>@bitbucket.org/<utilisateur>/<nom_du_dépôt>.git
(pour moi, c'était Lyyn@bitbucket.org/Lyyn)
```

3) Pour finir, il ne nous reste plus qu'à pousser tout notre dossier, ses commits, ... vers notre nouveau dépôt

```
> git push origin master
```

Notre mot de passe sera demandé interactivement, pas d'inquiétude

5. Résultat sur BitBucket

Une image vaut mieux que mille mots, on voit ci-dessous que la création du dépôt date d'il y a 18 heures mais que le premier commit date de bien plus longtemps. Preuve donc que ça a bien fonctionné

Cliquez sur l'image pour l'agrandir

Et pas de commentaire sur mon engrish, pitié. Ni de question sur la censure

6. Bonus : Perte du origin/HEAD

En effectuant ces manipulations, il se trouve que j'ai perdu le **"origin/HEAD"** qui sert à déterminer le dernier commit effectué. En soit, ce n'est pas bien grave, mais **SourceTree** se base dessus pour savoir combien de commit(s) doivent être push afin de garder à jour le dépôt distant.

Si jamais vous êtes dans le même cas que moi, retournez sur **Git Shell**, dans le dossier de votre nouveau dépôt et entrez la ligne de commande suivante :

```
> git symbolic-ref refs/remotes/origin/HEAD
    refs/remotes/origin/master
```

Ceci replacera notre **origin/HEAD** à la position de notre **origin/master**. Et **SourceTree** affichera à nouveau fièrement le nombre de commit(s) à push :

Voir : Une explication bien fournie par un utilisateur de Stack Overflow

---

Voilà donc la fin de ce petit guide qui me sera très certainement encore utile à l'avenir, et s'il peut servir à d'autres personnes, c'est tant mieux. Attention, je ne suis pas à l'abri d'une quelconque erreur ou coquille, si vous en voyez une ou que vous voulez apporter une précision, n'hésitez pas le signaler dans les commentaires.

Ce guide s'inspire **très fortement** d'un excellent billet de **AirBlade Software**, retrouvez-le à cette adresse : http ://airbladesoftware.com/notes/moving-a-subdirectory-into-a-separate-git-repository/ .

Je remercie également **Vayel** et **rz0** pour leur intervention sur ce site dans mon sujet de demande d'aide : http ://zestedesavoir.com/forums/sujet/2139/creer-un-nouveau-depot-a-partir-dun-dossier-en-conservant-lhistorique/ .

Et vive Git

7. Liens

[**FR**] Mon sujet de demande d'aide sur ZdS

[**EN**] Site de BitBucket
[**EN**] Site de GitHub
[**EN**] GitHub for Windows (+ Git Shell)
[**EN**] Atlassian SourceTree (GUI pour Git/Mercurial)
[**EN**] Billet de AirBlade Software