

Le coffre à jouets du petit git

version 1.3



I. Introduction

Linus Torvalds, créateur de Linux, sort la première version de **git** en avril 2005. Il s'agit d'un logiciel de gestion de versions. En d'autres termes, git permet de collaborer à plusieurs sur un même projet, tester, mettre à jour les fichiers et les versions en toute simplicité. Son fonctionnement décentralisé est à l'origine de son succès fulgurant à sa sortie.



Logo de git

Mais avant d'atteindre une telle appréciation dans le monde de l'informatique, git a d'abord été un sujet de controverse. Né à la suite de l'abandon de BitKeeper pour gérer le noyau Linux, la nouvelle création de Linus a été réalisée pour pallier tout ce qui ne lui convenait pas chez BitKeeper.

Le nom du projet n'a pas non plus fait l'unanimité, puisqu'il n'est rien de moins qu'un terme utilisé en argot britannique pour qualifier quelqu'un de « pourri ». A ce sujet, Linus dira d'ailleurs à PC World : « Je suis un bâtard égoïste, donc je donne mon nom à tous mes projets. D'abord Linux, maintenant Git. »*

Vous connaissez maintenant l'histoire de git. Certes, ça fait une bonne anecdote à raconter à ses amis ou pendant une réunion familiale, mais ça ne vous aidera pas à préparer votre projet du jour ! Au travail !

* « When asked why he called the new software, "git," British slang meaning "a rotten person," he said. "I'm an egotistical bastard, so I name all my projects after myself. First Linux, now git." », [source](#).

II. Consignes

- * En cas de question, pensez à demander de l'aide à votre voisin de droite. Puis de gauche. Demandez enfin à un Cobra si vous êtes toujours bloqué.
- * Si l'installation d'un logiciel se déroule mal, recommencez depuis le début.
- * Ce document retrace les commandes de base pour la gestion d'un dépôt – plus couramment appelé « repository » – sans collaboration avec un partenaire et sans gestion de branche. Pour en savoir plus sur ce qui peut être fait avec git, rendez-vous sur sa [documentation officielle](#).
- * Si vous savez utiliser git et que vous avez un compte sur un hébergeur de repositories, vous pouvez passer au projet Coding Club du jour dès maintenant.

III. Création d'un compte et d'un repository

a. Quel sera ton identifiant ?

Vous l'aurez peut-être compris, vous allez désormais vous servir de git pour gérer vos projets Coding Club. C'est un outil puissant qui vous sera très utile autant pour mettre à jour vos fichiers que collaborer avec d'autres personnes. Mais avant de s'en servir, vous devez avoir votre propre espace pour héberger vos projets. Nous allons donc utiliser :

GitHub

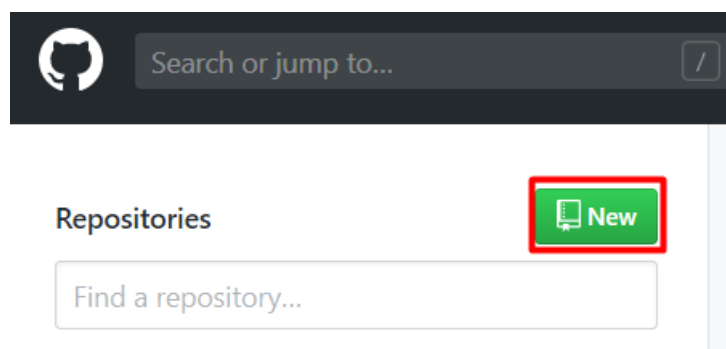
Logo de GitHub

Inscrivez-vous en ouvrant ce [lien](#) dans un nouvel onglet de votre navigateur. Nous utiliserons ce service car il est adapté aux débutants et possède une interface graphique. Petite astuce : pensez à composer un mot de passe fort avec une combinaison de plusieurs mots.

Vous avez donc un compte et êtes actuellement connecté dessus. Il est temps de créer un repository et de vous lancer dans vos projets !

b. Le premier repository

Une fois que vous êtes connectés, vous êtes sur la page d'accueil. Dans le coin supérieur gauche, vous trouvez ce bouton :



Partie de la page d'accueil de GitHub

Cliquez dessus et remplissez les informations suivantes. Le nom¹ doit être celui indiqué dans le sujet Coding Club du jour, la description² sera inscrite dans le « readme », petit fichier

explicatif du contenu du dépôt. N'oubliez pas de cocher la case⁴ pour qu'il soit automatiquement créé. Tu peux choisir de rendre public ou privé³ ton repository mais si tu veux le partager avec les Cobras et tes amis, il vaut mieux le garder visible. Il ne reste plus qu'à valider⁵ !

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository? [Import a repository.](#)

Owner

surnom ▾

/


Repository name *


1

Great repository names are short and memorable. Need inspiration? How about [shiny-meme?](#)

Description (optional)

2

☒  **Public**
Anyone can see this repository. You choose who can commit.


☐  **Private**
You choose who can see and commit to this repository.

3

Skip this step if you're importing an existing repository.

☒ **Initialize this repository with a README**
4 This will let you immediately clone the repository to your computer.

Add .gitignore: **None** ▾

Add a license: **None** ▾ 

Create repository

5

Page de création de repository de GitHub

Maintenant que vous avez créé votre repository, vous avez le choix entre utiliser une interface graphique ou des lignes de commande pour le gérer.

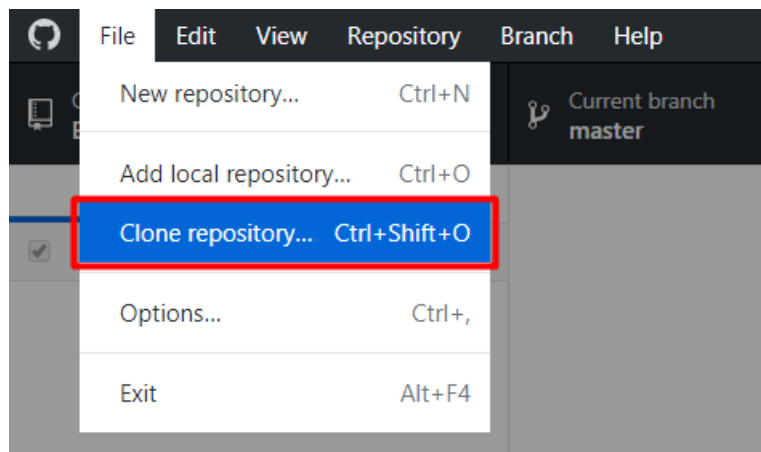
IV. Gestion d'un repository par une interface graphique

GitHub possède, entre autres, une interface graphique nommée GitHub Desktop. Vous pourrez la télécharger [ici](#).



Logo de GitHub Desktop

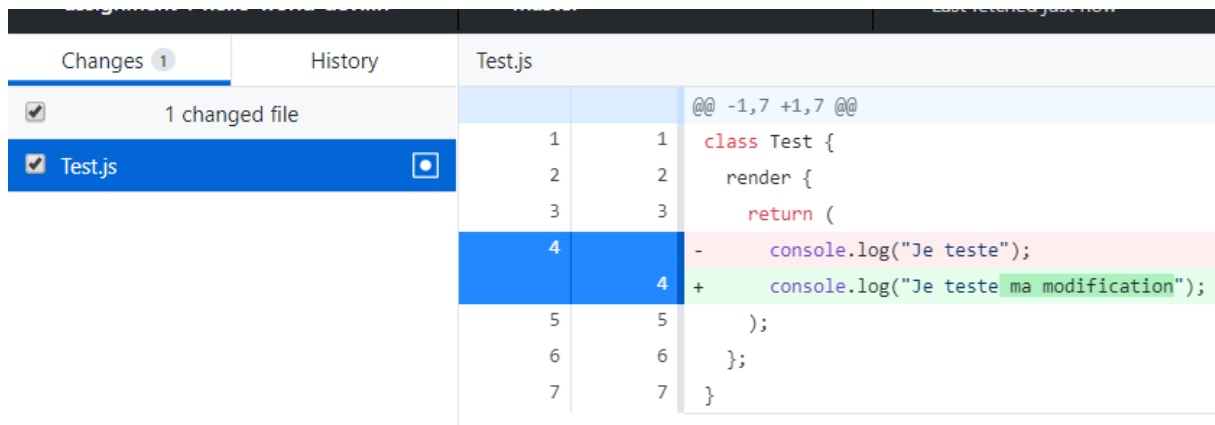
Lancez le fichier d'installation, suivez les étapes et connectez-vous dessus grâce au compte que vous venez de créer.



Menu de GitHub Desktop

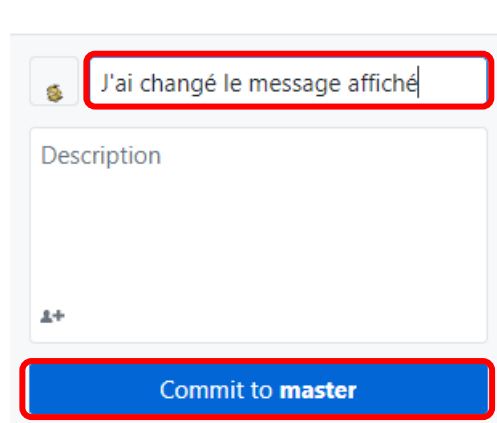
Dans l'onglet File du menu, vous pourrez cloner le repository que vous avez créé plus tôt. Le logiciel va donc installer une copie sur votre ordinateur pour que vous puissiez modifier des fichiers à distance, sans que ça impacte le repository sur votre espace GitHub.

Bien, maintenant vous commencez le sujet Coding Club du jour. Vous avez créé et modifié des documents. Il est temps désormais de les envoyer en ligne dans le but, entre autres, de sauvegarder votre travail et d'en faire profiter la communauté !



Page de changements dans les fichiers du repository ouvert sur GitHub Desktop

Dans l'exemple ci-dessus, un fichier apparaît dans la liste. L'affichage sur la droite indique quelle ligne – rouge – a été remplacée par une autre – verte. Si vous êtes satisfaits de votre réalisation, il est temps de préparer votre envoi en ligne.



Création d'un message de « commit » sur GitHub Desktop

A ce stade, vous écrivez un message explicite qui fait part de ce que vous avez modifié. Même si vous travaillez seul sur le projet, il est toujours préférable de pouvoir se repérer dans ce que vous faites. Vous pouvez alors cliquer sur « commit to master ». Le logiciel prend alors compte de l'état actuel de vos fichiers et crée une nouvelle version du repository. Cela peut s'apparenter à prendre vos modifications en photo dans l'attente de les imprimer pour les ajouter dans un album.

Quant à « master », il s'agit du nom de votre branche par défaut. Vous n'avez pas besoin de vous en occuper pour réussir ce tutoriel.

No local changes

There are no uncommitted changes in this repository. Here are some friendly suggestions for what to do next.



Push 1 commit to the origin remote

You have one local commit waiting to be pushed to GitHub.

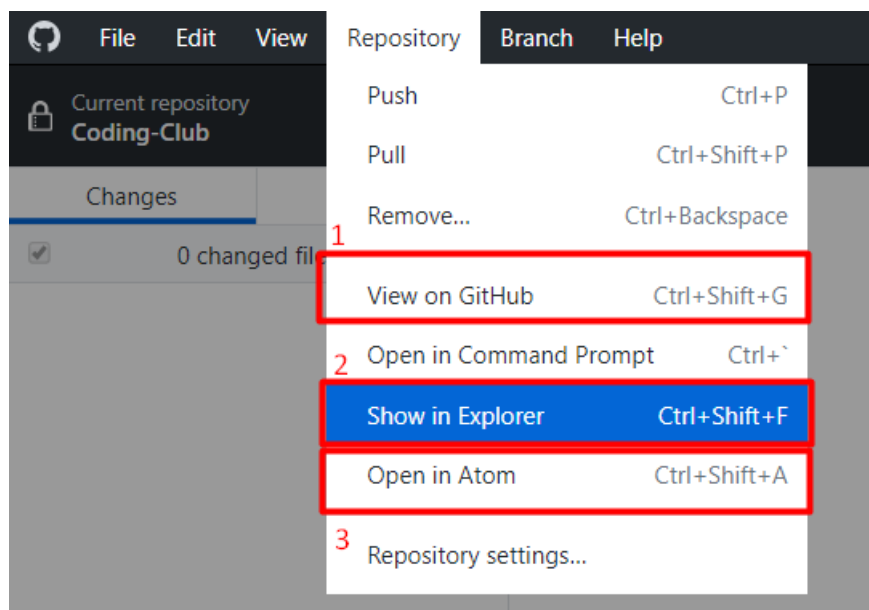
Always available in the toolbar when there are local commits waiting to be pushed or

Ctrl P

Push origin

Création d'un message de « commit » sur GitHub Desktop

Tout s'est bien passé pendant le stade de commit ? Parfait, vous pouvez maintenant envoyer votre photo à imprimer pour la ranger dans l'album. En d'autres termes, vous pouvez cliquer sur « push origin » afin d'intégrer vos changements au repository sur votre espace GitHub. Une fois la manipulation effectuée avec succès, regardez le contenu de votre repository dans votre navigateur. Tout est bien là. Vous pouvez continuer à réaliser votre projet !



Onglet « Repository » du menu de Github Desktop

Pour découvrir où se trouve votre dossier pour le manipuler manuellement, vous pouvez ouvrir le lien vers votre Github¹ ou dans votre explorateur de fichiers². Pour ouvrir votre repository dans un éditeur de texte³, assurez-vous d'en avoir relié un à votre logiciel. Dans l'exemple ci-dessus, Github Desktop est lié à Atom, un autre projet de la même compagnie.

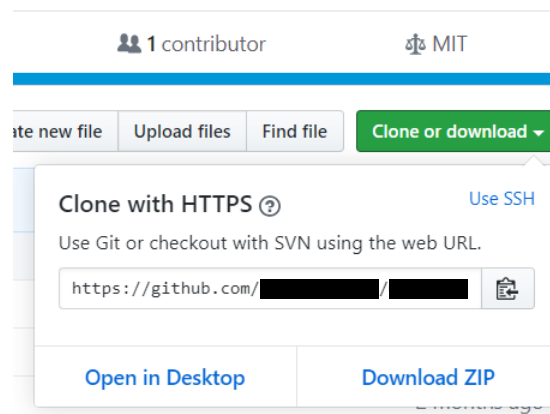
V. Gestion d'un repository par ligne de commande

Tout d'abord, pour ceux qui ne sont pas sur Linux, rendez-vous [ici](#) pour télécharger git et suivez les instructions du fichier d'installation. Pour les autres, installez-le grâce à votre gestionnaire de paquets.

Selon votre système d'exploitation, vous pourrez ensuite vous servir du :

- * PowerShell, en tapant son nom dans la barre de recherches Windows
- * Terminal Mac, en ouvrant Spotlight – Pomme + espace – et en cherchant « terminal »
- * Terminal Linux, situé dans vos applications

Maintenant, vous allez cloner le repository que vous avez créé sur votre compte GitHub.



Lien d'un repository sur GitHub

Copiez le lien **HTTPS** de votre repository et gardez-le de côté. Maintenant, dans votre terminal, dirigez-vous dans le dossier où vous souhaitez le stocker avec les commandes suivantes :

- * Pour afficher le contenu du dossier

```
$> ls
```

- * Pour se déplacer d'un dossier à un autre, utilisez la première commande. La deuxième permet de retourner au dossier parent

```
$> cd [nom du dossier]
$> cd ..
```

- * Pour créer un nouveau dossier

```
$> mkdir [nom du dossier]
```

Enfin, vous pouvez cloner votre repository avec la commande suivante. Vous pouvez ensuite créer des fichiers et les modifier pour réaliser le sujet Coding Club du jour.

```
$> git clone [lien HTTPS que vous avez copié]
```

Tout se passe bien, mais maintenant vous souhaitez envoyer vos fichiers sur le repository en ligne. La commande suivante vous permettra de voir dans quel état sont vos fichiers :

```
$> git status
```

Ceux qui ont été supprimés, modifiés ou créés seront visibles prioritairement. Vous savez maintenant quoi envoyer sur votre repository. Faites la manipulation suivante assez souvent afin de sauvegarder votre projet et de pouvoir le montrer à votre entourage. Remplacez bien le contenu entre crochets par vos données et description.

```
$> git add [nom du/des fichier(s)]  
$> git commit -m "[description des changements effectués]"  
$> git push origin master
```

Vous êtes un peu perdus ? La première commande indique que vous allez stocker des fichiers à part, en attente du « commit ». Celui-ci, dans la deuxième commande, sert à étiqueter le changement qui est en suspens. Enfin, le « push » envoie le tout à la branche du repository sur laquelle vous souhaitez voir vos changements apparaître. Par défaut, on utilise « master ».

Vous pouvez désormais poursuivre sereinement la création de votre projet. Si vous ne savez plus où vous en êtes, regardez vos messages de commit avec la commande suivante :

```
$> git log
```

Elle vous permet aussi de vous assurer que vos fichiers modifiés ont correctement été envoyés en ligne.

VI. Conclusion

Et voilà ! Vous avez un compte sur lequel vous avez créé le repository nécessaire pour votre projet Coding Club. Vous êtes désormais capable de le gérer pour continuer chez vous, générer de nouveaux projets personnels et les montrer à votre entourage.

Pour aller plus loin dans votre découverte de git, vous pouvez :

- * Vous renseigner sur le système de branches git
- * Découvrir comment mettre gratuitement en ligne un blog depuis un repository GitHub
- * Regarder comment vous pouvez contribuer à un projet open-source
- * Voir d'autres interfaces graphiques comme GitKraken ou ungit
- * Collaborer avec un-e camarade sur un projet
- * Vous intéresser aux tests unitaires et l'intégration continue

Mais ça, ça sera pour une prochaine fois ! Si vous êtes curieux, pensez à poser des questions à la fin de votre activité aux Cobras. Ils seront ravis de partager leurs connaissances avec vous !