

GitHub : Le réseau social des développeurs grâce à Git

Auteur: Hugo Ducommun

Date: 30 Mai 2019

GitHub est une plateforme de projets de jeunes développeurs motivés qui souhaitent publier leur travail de manière libre (OpenSource). *GitHub* est connu pour être pratique lorsqu'on travaille en équipe. Il permet à chaque collaborateur de travailler sur un seul et même projet sans influencer l'avancement des autres. Ce site web peut également être utilisé de manière professionnelle grâce à des comptes payants.

L'avantage de GitHub est qu'il est OpenSource. C'est à dire que même si nous ne sommes pas administrateur d'un projet, on peut tout de même y contribuer. En faisant une **pull request** (nous verrons cela plus tard) à l'administrateur pour qu'il examine notre idée et peut être la mettre en application sur son projet.

On parle souvent de git , de quoi s'agit-il ?

git est un logiciel de versioning (gestion de versions) que le site *GitHub* utilise.

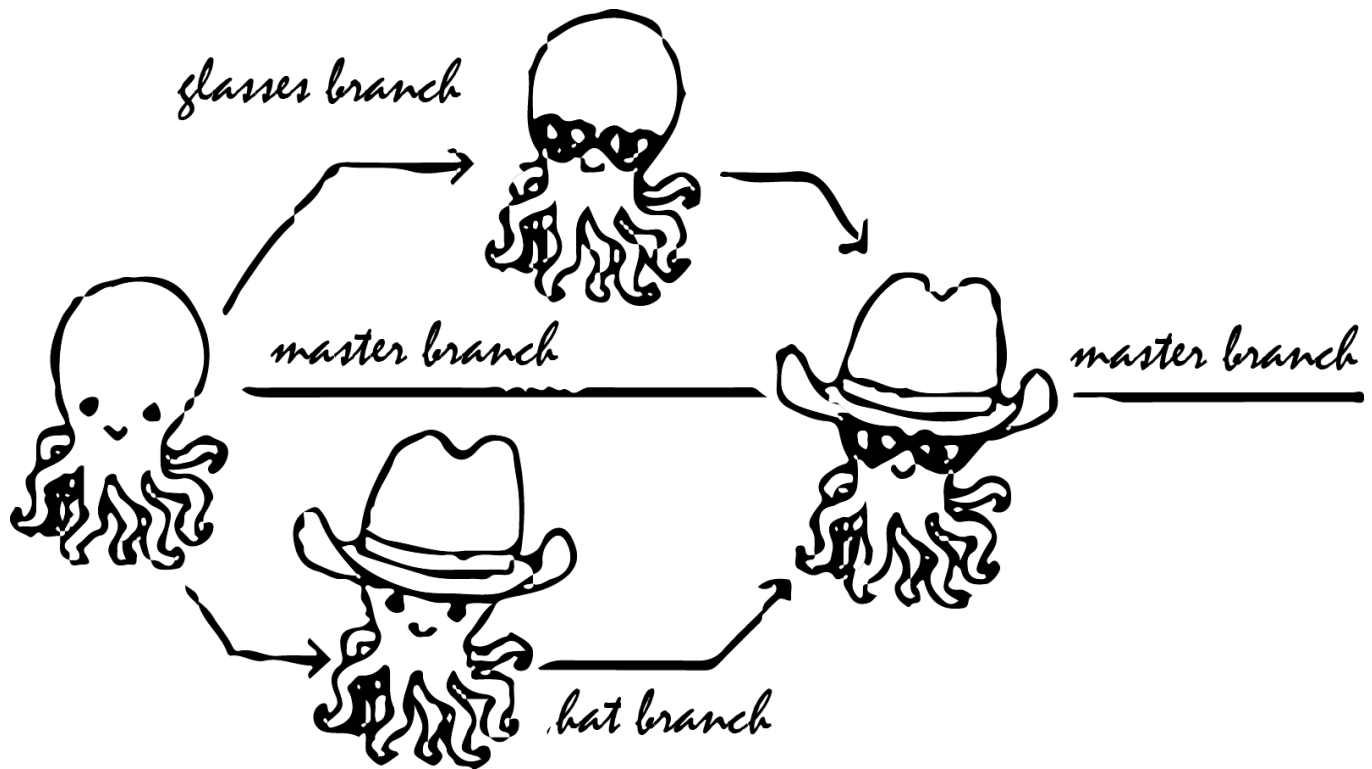
Il permet par conséquent de faciliter l'accès à l'historique des anciennes versions d'un projet et de synchroniser facilement les fichiers entre eux grâce à un système de **branches** que je développerai dans le point suivant.

En réalité c'est git qui est à la base du site web *GitHub*. Mais *GitHub* a rajouté une interface graphique à git qui s'utilise principalement dans un terminal, c'est pour cela que *GitHub* est plus connu que le logiciel de versioning lui-même. Pour cette raison, j'étudierai ici principalement le logiciel git et rajouterai quelques informations supplémentaires sur *GitHub*.

Fonctionnement de git

Introduction imagée de la notion de *branch*.

Git procède en branches. Voici un petit schéma que j'ai trouvé très expressif sur la manière dont ça fonctionne.



Nous avons donc deux types de *branches* différentes, oui oui deux et pas trois comme sur le schéma. Il y a la *branch* du milieu appelé **master branch**, c'est là où se situera la version officielle et fonctionnelle de votre projet. Puis un deuxième type appelé **feature branch** qui est caractérisé par les branches nommées sur le schéma, hat et glasses (en réalité ce sont toutes les branches exceptés la master branch).

Le fonctionnement est simple, le projet est ici de rajouter un chapeau et des lunettes à l'image du poulpe de base. Un collaborateur s'occupera donc du chapeau (C1) et un autre des lunettes (C2). Ils vont procéder de cette manière :

1. C1 et C2 vont donc copier le projet actuel (master branch) dans une feature branch personnelle.
2. Faire leurs modifications et les rendre fonctionnelles (ajoutez un chapeau ou des lunettes).
3. Uploader leurs modifications dans la master branch pour avoir un projet complet.

Termes techniques

Bien sûr, ceci est un peu plus compliqué dans un vrai projet, il y a beaucoup plus de choses à faire que rajouter deux accessoires, et nous devons procéder par ligne de commandes. Mais c'est une bonne approche de cette notion.

Je vais donc ici détaillé quelques termes qu'utilise git dans son fonctionnement.



Repository

Un repository (en français dépôt) est l'ensemble de votre projet : les documents que vous éditez et dont vous suivez les modifications s'y trouvent. Le repository peut être locale ou se trouver sur votre serveur dédié.

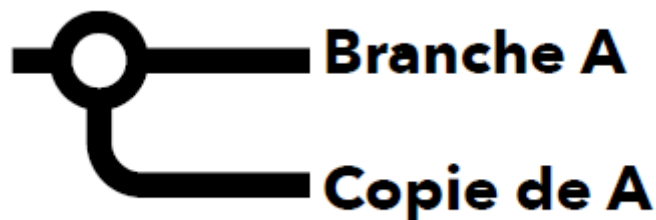
Branch

C'est une des branches copiées de la master branch par défaut. Après avoir créé la branche copiée de master, elle ne sera plus affectée par les changements opérés sur les autres branches du projet.

Sur le schéma ci-dessous, 'Copie de A' est une branche de 'Branche A'. D'ailleurs pour avoir copié la branche A, l'utilisateur a dû **merge**.

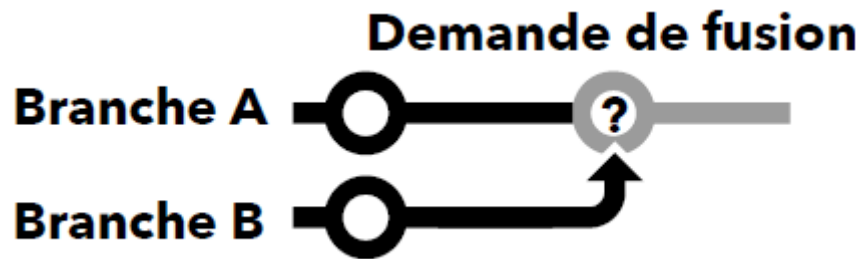
La commande pour créer une nouvelle branche est : `git branch nomNouvelleBranche`

La commande : `git branch --list` vous permet de voir la liste de toutes les branches du repository actuel.



Pull request

Ce terme peut être traduit par demande de fusion (**merge**). C'est lorsque le collaborateur veut fusion sa branch avec une autre (généralement la master branch) pour appliquer les changements tels que les corrections de bugs ou ajout de fonctionnalité à la branche cible. Le responsable de la branche ciblée est libres de refuser ou accepter ce **pull request**.



Fork

Fork (littéralement fourchette en français), correspond à copier une branche déjà existante. On fork souvent la master branch au début pour pouvoir se créer notre propre branch et modifier le projet sans impacter sur la master branch.



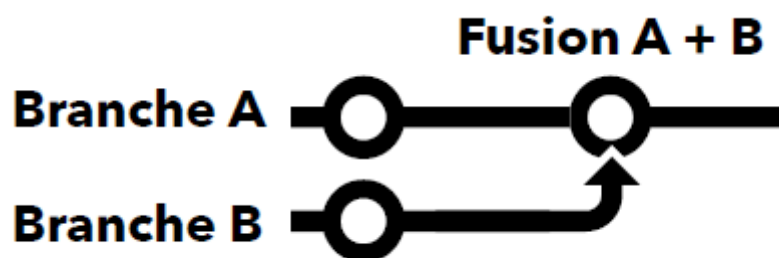
Merge

Merge est un peu le contraire de fork, après avoir modifié tout ce que l'on voulait, on peut fusionner notre branche avec une autre. Cette fonctionnalité est souvent protégée par un pull request sinon tout le monde pourrait modifier n'importe quelle branche.

Les modifications de la branche B seront donc appliquées à la branche A si le merge fonctionne.

La commande pour fusionner la branche B est : `git merge brancheB`

Attention il est important d'exécuter cette commande depuis la branche A !

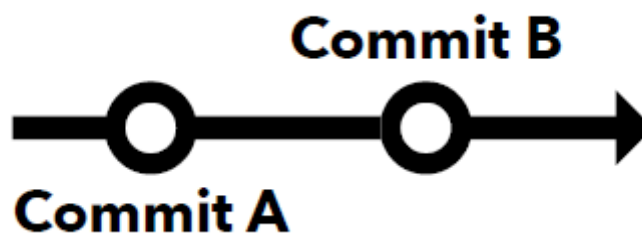




Commit

Commit est l'action la plus courante que vous allez exécuter avec git. Elle correspond, comme l'indique son icône à une modification de la branche en question. Lorsque vous avez localement modifié une branche, vous devez **commit** pour enregistrer les changements, généralement avec un message d'information pour pouvoir par la suite mieux retrouver des anciennes modifications.

La commande pour commit en rajoutant un message d'exemple est : `git commit -m 'Add the cow-boy hat'`



Push

Envoyez tous vos commits dans le serveur dédié sur lequel est hébergé le repository (dépôt distant). Vous 'envoyez' en quelque sorte vos fichiers à vos collaborateurs.

La commande pour push est : `git push`



Pull

Effet contraire de push, vous recevez les fichiers envoyés par vos collègues. Avant chaque grosse séance de travail, assurez-vous de pull pour voir l'avancement de votre équipe. Il charge les dossiers et fichiers du repository sur votre machine en local.

La commande pour pull est : `git pull`

Autres commandes dans git

Nous avons déjà vu quelques commandes dans les termes techniques, voici le reste des commandes basiques :

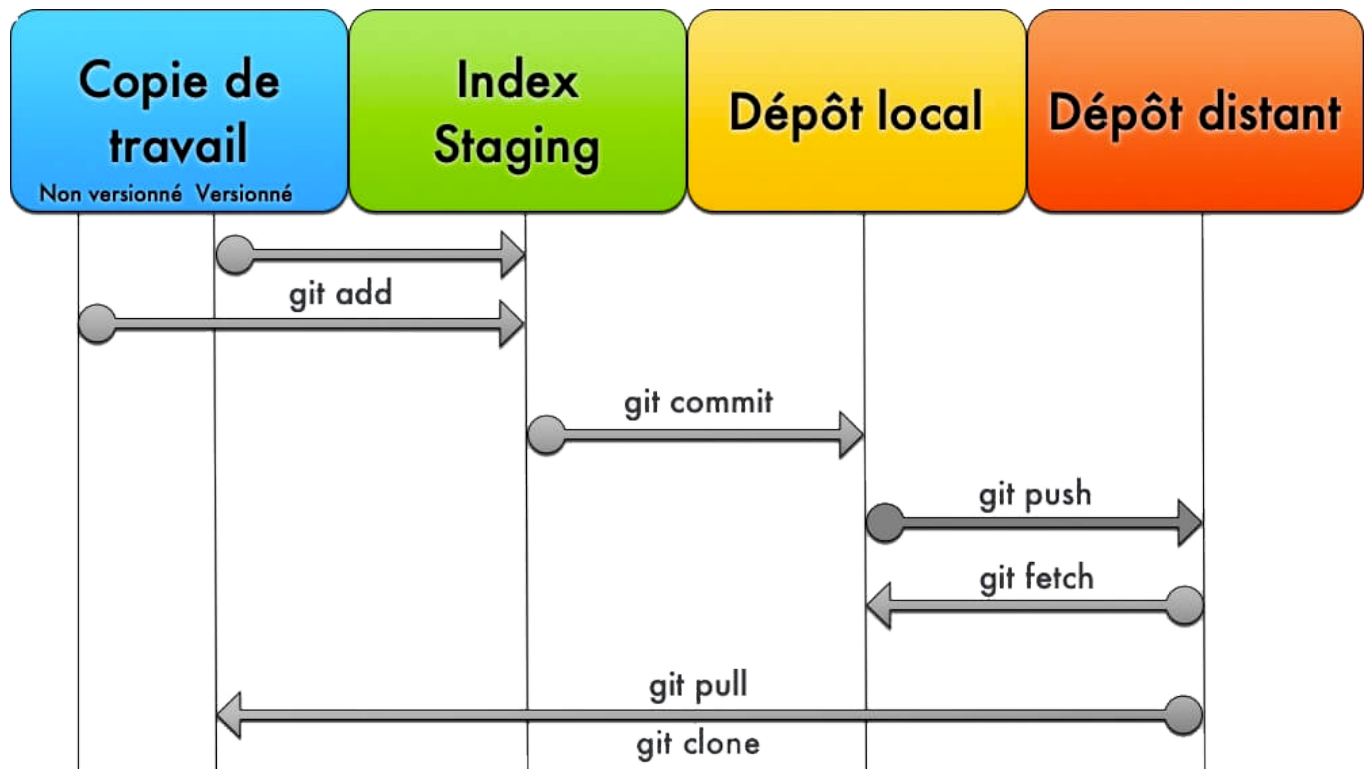
- `git init` : initialise votre dossier en tant qu'un dossier git
- `git clone URL` : clone un repository déjà existant dans le dossier depuis lequel vous exécutez la commande
- `git status` : affiche le statuts des fichiers de votre repository. Permet de voir où nous en sommes.
- `git add nomFichier` : ajouter des fichiers dans l'index (pré-sauvegarde). La commande `git add *` ajoute tous les fichiers modifiés.
- `git checkout nomBranche` : s'utilise en tant que switch d'une branche à une autre (utilisation basique)

Lors de la première utilisation de git, vous devrez enregistrer votre pseudo et votre email. Après avoir utilisé la commande `git init` qui initialise votre dossier en tant qu'un dossier git, utilisez les deux commandes ci-dessous :

- `git config --global user.name 'hugoducum'`
- `git config --global user.email 'prenom.nom@bugnon.educanet2.ch'`

L'avantage de Git reste que c'est une source particulièrement bien documentée en ligne car ceux qui le maîtrisent sont en général assez actifs sur les forums. On arrive toujours à trouver de l'aide sur les différentes plateformes. Aussi grâce aux commandes : `git help nomCmd` par exemple `git help checkout`.

Schéma récapitulatif de git et ses commandes



La commande `git fetch` ne sera pas traitée ici.

Publication d'un fichier

En somme lorsqu'on veut publier un fichier, par exemple *index.html* dans notre repository, il faut taper dans l'ordre :

1. `git pull`
2. `git add index.html`
3. `git commit -m 'Ajoute de mon fichier html'`
4. `git push`

Le `git pull` du début permet de ne pas avoir de conflit de fichier lorsqu'on push en mettant à jour notre copie de travail versionnée.

Cloner un repository déjà existant

C'est souvent ce que vous allez faire lors de projet GitHub, le projet sera déjà crée et il vous faudra le cloner dans un dossier. Grâce à la commande `git clone URL` l'URL est l'endroit où est stocké le repository, ça peut par exemple être sur GitHub sous la forme <https://github.com/Bugnon/oc-2018.git> (<https://github.com/Bugnon/oc-2018.git>).

La commande checkout

Elle vous permet comme expliqué plus haut, de switch entre les branches sur lesquelles vous travaillez.

Elle est aussi utile sous la forme `git checkout -- .` qui vous permet d'annuler les changements que vous avez fait dans votre fichier localement. Attention, ça ne marchera plus après un commit.

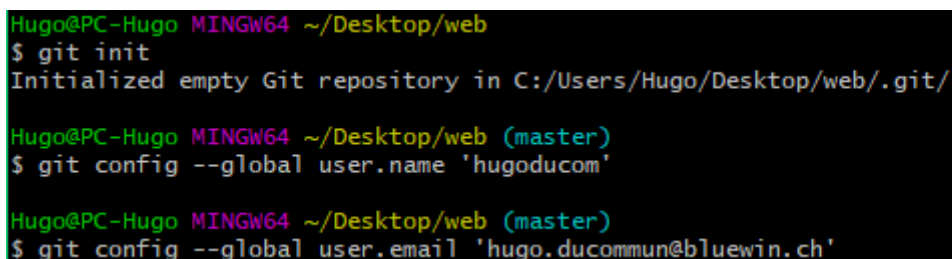
Exemple pratique d'une création d'un repository de A à Z

Je suis un jeune codeur web qui souhaite partager mes premiers pas sur une plateforme de développement comme GitHub. Je crée donc un dossier sur mon bureau appelé "web". C'est ce dossier que je souhaite partager sur GitHub. Je télécharge donc [Git](https://git-scm.com/downloads) (<https://git-scm.com/downloads>).

Il s'agit d'un petit projet, je travaillerai donc seulement dans la *master branch* et ne créerai pas d'autre branche.

Après l'installation, je fais clique-droit sur mon dossier et appuie sur **Git Bash Here**. Une console s'ouvrira et c'est depuis là que vous taperez vos commandes.

Comme je me suis informé sur ce notebook, je tape d'abord `git init`. Et enregistre mes informations à l'aide de `git config`.



```
Hugo@PC-Hugo MINGW64 ~/Desktop/web
$ git init
Initialized empty Git repository in C:/Users/Hugo/Desktop/web/.git/

Hugo@PC-Hugo MINGW64 ~/Desktop/web (master)
$ git config --global user.name 'hugoducum'

Hugo@PC-Hugo MINGW64 ~/Desktop/web (master)
$ git config --global user.email 'hugo.ducumun@bluewin.ch'
```

Je commence ensuite à développer tranquillement dans ce dossier. Je crée donc mon fichier *index.html*. Je commence à coder et arrive le moment où je souhaite mettre en ligne ce que j'ai fait. Je fais donc un `git add index.html` ou `git add *` si je veux add tous les fichiers de mon dossier web.

Je remarque en utilisant la commande `git status` que mes fichiers sont prêts à être commit au dépôt local.

Puis `git commit -m 'Ajout de la première version de mon site'`.

Astuce : Si je me suis trompé et que je remarque après le `git add index.html` mais avant le `git commit` et le `git push`, je peux utiliser la commande `git rm --cached index.html` (juste `index.html`) ou `git rm -r --cached .` (tous les changements) pour retirer les changements de l'index staging (schéma ci-dessus), cette commande est en quelque sorte l'inverse de `git add`.

Attention, le `-r` que j'ai rajouté est une option qui signifie récursivement. C'est-à-dire si j'ai des dossiers avec des fichiers à l'intérieur, les fichiers intérieurs seront également retirés.

```
Hugo@PC-Hugo MINGW64 ~/Desktop/web (master)
$ git add index.html

Hugo@PC-Hugo MINGW64 ~/Desktop/web (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   index.html

Hugo@PC-Hugo MINGW64 ~/Desktop/web (master)
$ git commit -m 'Ajout de la première version de mon site'
[master (root-commit) 3b9c638] Ajout de la première version de mon site
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 index.html
```


Rendez-vous maintenant sur [GitHub \(https://github.com/new\)](https://github.com/new) pour créer notre repository. Je me connecte et remplis les informations nécessaire.

Create a new repository


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

Owner

Repository name *

 hugoducum

 /

web 

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

Description (optional)

My first test with git and github

☒ Public

Anyone can see this repository. You choose who can commit.

☐ Private

You choose who can see and commit to this repository.


☐ Initialize this repository with a README

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None

 |

Add a license: None



Create repository

Par la suite il faut exécuter les deux commandes que GitHub nous demande :

- `git remote add origin https://github.com/hugoducum/web.git`
- `git push -u origin master`

Il se peut que lors de la deuxième commande, on vous demande votre login et votre mot de passe GitHub. Il faut donc avoir un compte GitHub.

```
Hugo@PC-Hugo MINGW64 ~/Desktop/web (master)
$ git remote add origin https://github.com/hugoducum/web.git

Hugo@PC-Hugo MINGW64 ~/Desktop/web (master)
$ git push -u origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 236 bytes | 236.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/hugoducum/web.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

Le plus dur est fait ! Votre repository est en ligne sur GitHub, bravo ! En rechargeant la page <https://github.com/hugoducum/web> (<https://github.com/hugoducum/web>), vous allez tomber sur votre fichier *index.html*.

My first test with git and github

Edit

Manage topics

1 commit

1 branch

0 releases

0 contributors

Branch: master

New pull request

Create new file

Upload files

Find File

Clone or download



hugoducum Ajout de la première version de mon site

Latest commit 3b9c638 12 minutes ago

index.html

Ajout de la première version de mon site

12 minutes ago

Help people interested in this repository understand your project by adding a README.

Add a README

Pour la suite de votre aventure de développement web, il vous faudra simplement suivre le point 'Publication d'un fichier' un peu plus haut de ce dossier.

Lorsque vous aurez compris le principe de git et serez capable de tout faire en lignes de commande, vous pourrez télécharger des applications qui feront le travail à votre place comme [GitHub Desktop](https://desktop.github.com/) (<https://desktop.github.com/>), qui facilitera grandement vos partages de fichier dans votre carrière de développement.

Sources:

- <https://gerardnico.com/code/version/git/branch> (<https://gerardnico.com/code/version/git/branch>)
- <https://fr.wikipedia.org/wiki/GitHub> (<https://fr.wikipedia.org/wiki/GitHub>)
- <https://fr.wikipedia.org/wiki/Git> (<https://fr.wikipedia.org/wiki/Git>)
- <https://www.sebastien-gandossi.fr/blog/difference-entre-git-reset-et-git-rm-cached> (<https://www.sebastien-gandossi.fr/blog/difference-entre-git-reset-et-git-rm-cached>)
- <https://www.youtube.com/watch?v=4o9qzbssflI> (<https://www.youtube.com/watch?v=4o9qzbssflI>)
- T rence Chevroulet pour les sch mas des commandes Git