Git & GitHub
Activité Partie 3

Table des matières

Qu'est-ce qu'un commit :

Pour simplifier la chose, dites-vous qu'un commit est un peu comme faire une sauvegarde d'un fichier Word, Excel, txt,... ou tout autre document pour lequel vous auriez fait un simple CTRL-s.

« *Soit, mais alors pourquoi s'embêter avec GIT, si ce n'est rien de plus que ça ?* »

Et bien la grande différence réside dans le principe du versionning, c'est-à-dire, de conserver une copie des anciennes sauvegardes, afin de pouvoir y accéder de nouveau si nécessaire.

Contrairement à des logiciels de traitement de texte comme Word, lorsque vous faite une sauvegarde, l'ancienne version de votre fichier est écrasée par la nouvelle, sans aucun moyen de pouvoir y accéder de nouveau, à moins de faire une copie à part entière de votre document à chaque sauvegarde, ce qui pourrait devenir compliqué à gérer, si vous deviez créer un document conséquent demandant un grand nombre de mise à jour, de travail et donc de sauvegarde.

En bref, un commit est une sauvegarde, ou, contrairement à d'autres logiciels simples, GIT permet de conserver une trace de chacune d'elle, vous donnant également la possibilité d'y naviguer et ainsi de retrouver l'état de vos fichiers à l'instant 'T' du commit.

Voici la commande « git commit -m "monCommentaire" »

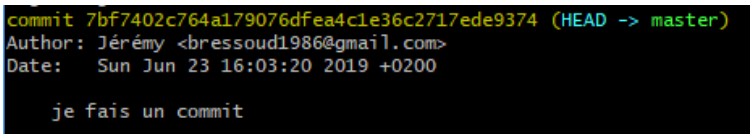
À quoi sert la commande git log :

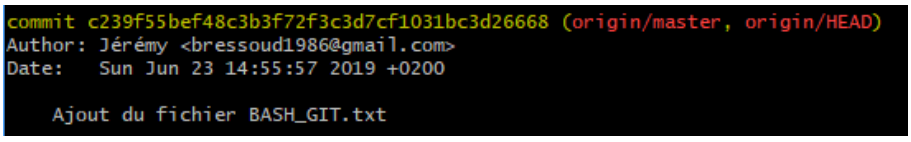
Vous rappelez-vous de ce que je vous ai expliqué pour le commit ?
Et bien la commande « git log » permet justement de visualiser la liste de vos commits.

« *OK, bon, imaginons que j'utilise GIT et fais des commits pour le suivi de mes sauvegardes, mais que j'en ai des dizaines et des dizaines, je fais comment pour m'y retrouver dans cette liste?* »

Et ben c'est simple, au moment de faire votre commit, la dernière partie est réservée à votre commentaire, ce qui vous permet d'être précis concernant la raison de ce commit.

Exemple, la dernière ligne de chacune de ces captures étant votre commentaire, lequel de ces commits vous parle le mieux :





Il n'y a pas photo, si ?

De plus, chacun de votre commits sera constitué des infos suivantes :

- SHA unique (la longue ligne de caractères jaune imbouffable à droite de commit)
- Les infos sur l'auteur dû commit, avec nom et adresse mail
- L'horodatage du commit
- Votre commentaire

Du coup, du moment que vos commentaires restent simples, concis, clairs et précis sur les modifications apportées à votre commit, vous pourrez non seulement toujours vous y retrouver, mais ce sera également beaucoup plus simple et sûr pour vous.

Qu'est-ce qu'une branche :

Bon, pour celui-là, j'ai dû sortir mon plus beau paint.net « ne me tapez pas s'il-vous plaît ;P ».

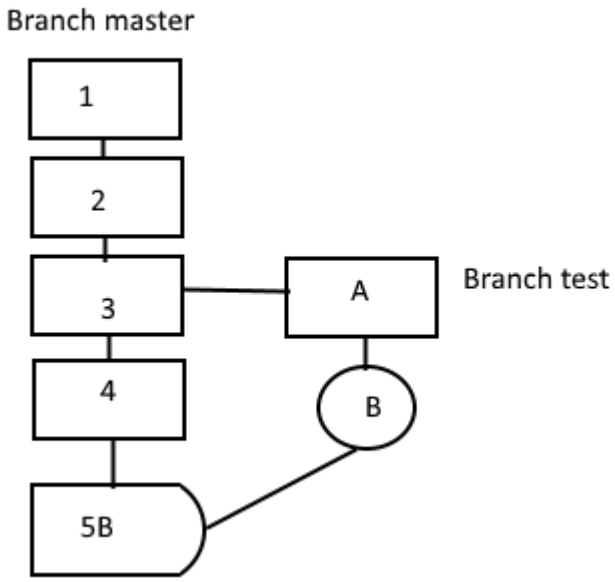
On va partir du principe que vous avez compris le principe des commits.

La branche master correspond au dossier principal de votre projet, et contient les éléments que vous allez commités.

Seulement, il peut arriver que pour une raison x y vous aillez besoin de tester quelque chose dont vous n'êtes pas forcement sûr du résultat.
Cependant, vous ne souhaitez pas corrompre votre fichier avec ceci, ci par tout hasard, ce test ne fonctionnerai pas.

Et ben pour cela, vous pouvez créer une seconde branche « Branch test » dans cet exemple, qui va créer une copie exact de votre branche master, puis par la suite, va la rendre totalement indépendante.

Grâce à cela, vous pourrez faire les tests que vous souhaitez sans prendre le risque de corrompre votre travail initial.



Cette branche fonctionnera de la même manière que votre branche master, vous pourrez y ajouter les dossiers, fichiers ou éditer vos documents comme bon vous semble, toutes ces modifications resteront dans la branche test.

Cependant, vous pourrez aussi par la suite :

- Synchroniser les informations des 2 branches, si vos tests sont concluants
- Conserver cette branche test indépendante, afin d'en faire un simple bac à sable pour vos expériences
- Supprimer cette branche test, si d'aventure celle-ci ne vous servez plus

Pour les commandes :

- git branch = vous permet de voir les branches existantes de votre projet
- git branch maBranch = vous permet de créer une nouvelle branch « maBranch »
- git checkout nomBranch = vous permet de naviguer dans les branches