# EXPLICATIONS GIT

[EXPLICATIONS GIT 1](#_Toc500169058)

[1 - Qu'est-ce qu'un commit ? 1](#_Toc500169059)

[2 - À quoi sert la commande git log ? 1](#_Toc500169060)

[3 - Qu'est-ce qu'une branche ? 2](#_Toc500169061)

[4 – Aller plus loin 2](#_Toc500169062)

## 1 - Qu'est-ce qu'un commit ?

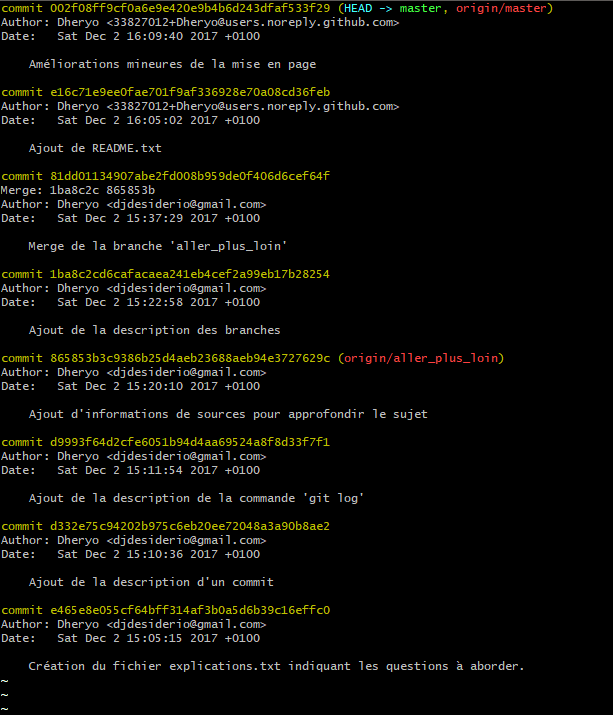
Un commit c'est comme une photo de l'état de ton projet. Quand tu as fini de modifier une partie de ton projet (correction d'une fonction par exemple), tu fais une 'photo' (un commit) qui garde une image de l'état actuel du projet. N'oublie pas d'indiquer un message explicite qui correspond à tes modifications ("Correction de la fonction 'nom\_de\_la\_fonction'" par exemple). C'est important pour t'y retrouver plus tard et/ou travailler à plusieurs sur le projet.

Ensuite, tu peux continuer à travailler sur ton projet et consulter à tout moment les différents commits ou revenir sur l’un d’eux. C’est très pratique pour organiser ton travail, revenir facilement en arrière si besoin et pour travailler à plusieurs. Fais autant de commits que nécessaire mais pas pour rien histoire de ne pas alourdir la liste de commits inutilement. J'insiste sur le fait de pouvoir s'y retrouver plus tard. C'est comme quand tu as appris à respecter des conventions d'écriture dans tes programmes et à les commenter clairement.

## 2 - À quoi sert la commande git log ?

La commande ‘git log’ sert simplement à afficher la liste de tous les commits effectués sur la branche active de ton projet (je reviens sur les branches après puisque c'est ta dernière question).

Voici un extrait de log ci-dessous :

Les commits sont affichés du plus récent au plus vieux avec comme informations :

- le SHA (son identifiant unique qui te sert notamment à revenir sur ce commit),

- le nom de l'auteur,

- la date et l'heure du commit,

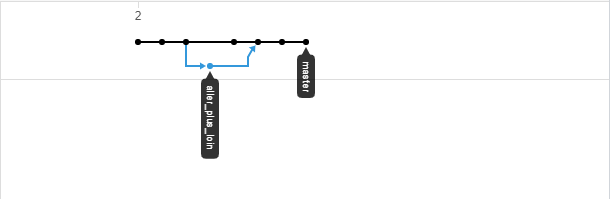
- le fameux message de description du commit qui doit être le plus clair possible,

- des informations entre parenthèses liées à la synchronisation du projet local et du repository, je ne détaillerai pas.

## 3 - Qu'est-ce qu'une branche ?

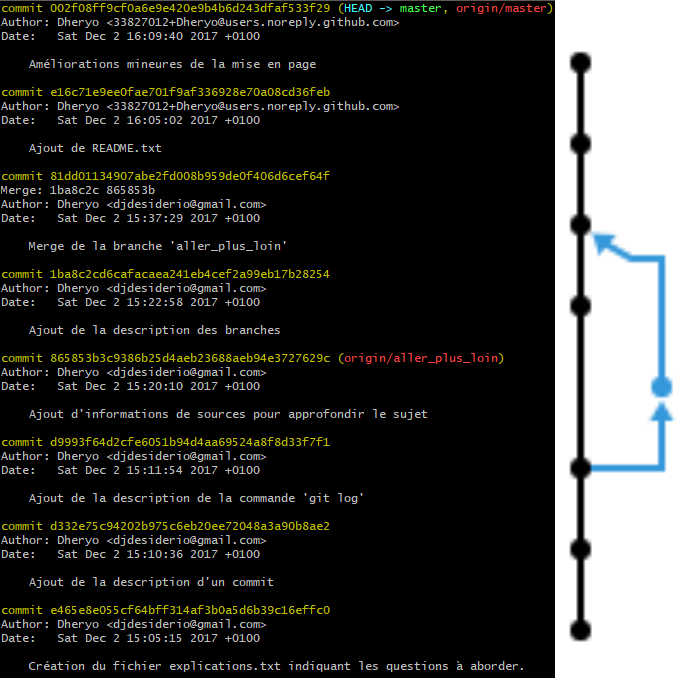
Il me semble qu'un exemple d'utilisation des branches sera plus parlant. Admettons que tu développes un jeu solo. En cours de développement, tu te dis qu'il faudrait essayer d'ajouter un mode multi. Plutôt que travailler sur la même version du projet, tu vas créer une nouvelle branche. A partir de là, les commits que tu vas faire sur la branche sobrement nommée "multi" et les commits que tu feras sur la branche par défaut qui s'appelle "master" seront indépendants.

Tu auras donc un projet avec deux versions parallèles comprenant les commits effectués avant la création de la branche "multi" suivis de commits indépendants. Ensuite, tu pourras 'merger' les deux branches (c’est-à-dire les fusionner) ou la supprimer si ton mode multi ne s'intègre pas dans ton jeu ou encore laisser la branche "multi" de côté en attendant de décider si elle devra être intégrée ou non.



Ci-contre une représentation de projet avec deux branches. Elle a été générée sur GitHub et reprend la branche principale « master » en noir et une seconde branche « aller\_plus\_loin » en bleu.

On peut ainsi corréler les deux images entre elles de la façon suivante :



On voit ainsi :

* que chaque point est un log,
* qu’après le 3ème commit, a été créée une nouvelle branche nommée « aller\_plus\_loin »,
* que le 4ème commit du log a été fait dans cette branche,
* que le commit suivant a été effectué sur la branche « master ». **A ce stade, chaque branche contient 4 commits, mais le dernier commit de chaque branche est différent,**
* que le commit d’après consiste à ré-unifier les deux branches, ce qu’on appelle un « merge »

## 4 – Aller plus loin

Tu peux entre autre, consulter [la documentation officielle](https://git-scm.com/documentation), le [cours Openclassrooms](https://openclassrooms.com/courses/gerer-son-code-avec-git-et-github) ou aller sur [le repository GitHub](https://github.com/Dheryo/explications_git) que j’ai créé pour l’occasion. En espérant avoir été suffisamment clair. Bonne continuation !