Gérez votre code avec Git et GitHub Activité 3

Boukary OUEDRAOGO

4 mai 2019

Consignes

L'objectif de l'activité est d'expliquer Git à une personne qui ne l'a jamais utilisée au travers des réponses aux questions suivantes :

1. Qu'est-ce qu'un commit ;
2. À quoi sert la commande git log ;
3. Qu'est-ce qu'une branche.

Dans ce qui suit nous donnerons des éléments de réponses à ces questions afin de permettre à un débutant de comprendre le fonctionnement de Git

---

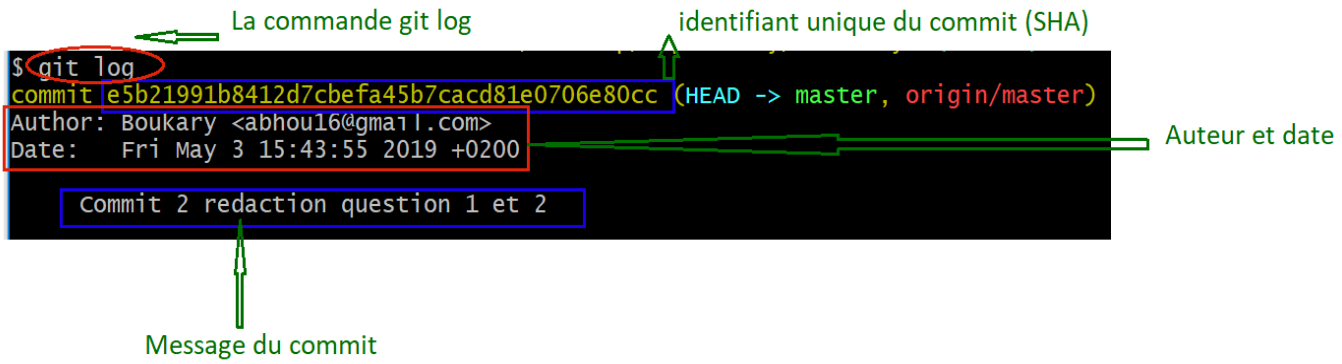
Question 1 : Qu'est-ce qu'un commit ?

En informatique, qu'il s'agisse d'un projet personnel ou d'un projet multi-colaborateurs, il est nécessaire de bien garder les traces des différentes modifications apportées au projet. Cela s'appelle du versionning dans le langage du commun des informaticiens. Pour réaliser ce versionning de code de manière claire et cohérente, Linus Torvald à crée un outil puissant appelé Git.
Git permet de suivre les modifications de tous les fichiers indexés dans un projet. Indexer un fichier dans Git consiste à signaler à Git que l'on souhaite que le fichier en question fasse partir de ceux que l'on souhaite suivre les modifications. Cela se fait au travers de la commande git add *nomDuFichier.extension*.
Après avoir modifié les fichiers du projet et les ajouté à l'index de Git, il faut valider les modifications et ajouter un message en guise de commentaire pour permettre de se retrouver plus facilement et de savoir précisement les modifications qui ont été faites. En d'autres termes il s'agit de sauvegarder la version courante du projet. Pour réaliser cela on utilise un commit.
Un commit n'est rien d'autre qu'une sauvegarde d'une version d'un projet avec des commentaires indiquant les modifications qui ont été faites dans cette version. La commande git pour faire un commit est :git commit -m "*commentaire* " .
Pour illustrer mon propos je prends l'exemple mon cas lorsque je rédigais cette note. Quand j'ai fini de rédiger les réponses aux deux premières questions, j'ai ajouté le ficher à l'index de git en faissant :
git add Activity3.pdf
Ensuite j'ai fait un commit pour expliquer un peu les modifications que j'ai apportées en faisant :
git commit -m "Commit 2 blackaction question 1 et 2"
En faisant cela j'ai crée une version de mon projet.

Question 2 : À quoi sert la commande git log ?

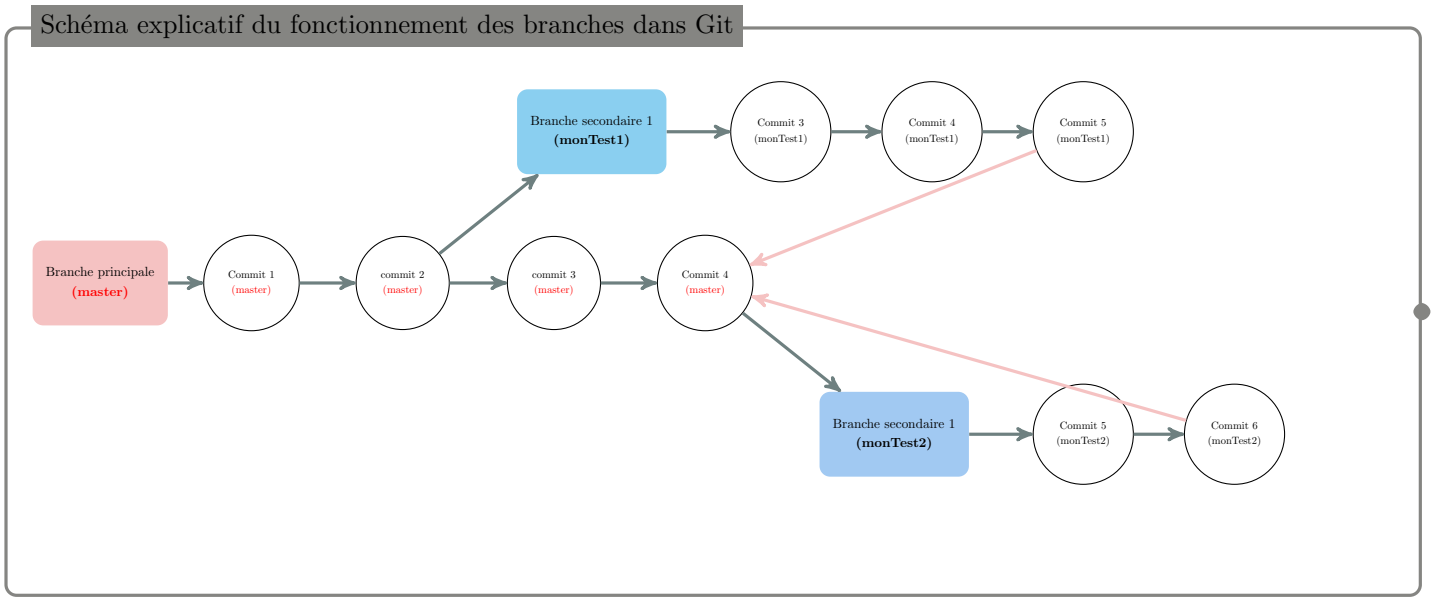
Lorsqu'on réalise plusieurs commits dans un projet, il arrive que l'on veuille voir l'historique des modifications qui ont été faites, en d'autres termes, des différentes versions du projet. Cela est nécessaire surtout quand il s'agit d'un projet multicolaborateurs où l'on veut savoir qui a fait telle ou telle modification. C'est dans cette situation qu'intervient la

commande git log.
La commande git log permet d'afficher l'historique des commits, de savoir précisement qui a fait tel ou tel commit et les messages précisant les modifications. En effet la commade git log affiche tous les commits, avec pour chacun d'eux son identifiant unique appelé SHA, le nom de l'auteur du commit et son adresse e-mail, la date du commit ainsi que le commentaire que l'auteur a choisi pour son commit.
La syntaxe de la commande Git log est tout simplement : git log.
En guise d'illustration voici le résultat du git log que j'ai effectué après le précédent commit.



Question 3 : Qu'est-ce qu'une branche ?

La notion de branche est fondamentale lorsque que l'on travaille avec git. A partir de la branche principale qui est par défaut master et qui contient les premiers commits, on peut créer d'autres branches qui contiendront d'autres commits. Cette façon de procéder est souvent nécessaire lorsque l'on souhaite faire des tests sans pour autant les appliqués à la branche principale. Illustrons cela par un schéma explicatif



Comme le montre la figure, à partir du deuxième commit sur la branche principale on a crée une première branche secondaire. Le but est de faire des tests sur cette branche sans altérer la branche principale. Une fois que nous sommes sûr de ce que nous avons fait sur cette branche secondaire, nous pouvons ajouter les commits qui y sont réalisés sur la branche principale. Cela ce fait au moyen de la commande git merge. Pour créer une nouvelle branche, il faut d'abord se positionner sur la branche à partir de laquelle la nouvelle branche doit être créée à l'aide de la commande suivante : git checkout *nomDeLaBranche*.
Pour afficher les différentes branches du projet on utilise la command :git branch. La branche active c'est-à-dire celle sur laquelle nous sommes en train de réaliser les commits sera affichée en vert
Si je veux ajouter les commits de ma branche monTest1 à la branche principale master, je me positionne sur la branche master en faisant git checkout *master* ensuite je fais git merge *monTest1*