Gérer son code avec Git et GitHub

Claude BUENO

Collègue développeur, je viens te parler d'un moyen efficace de suivre l'évolution de ton code sans rien perdre des modifications que tu as pu faire tout au long de la vie de ton projet. Le parcours de formation Développeur de OpenClassrooms m'a permis de faire mes premiers pas avec Git et Github. Aussi, je voulais partager avec toi cette pépite.

Git est un logiciel développé par Linus Torvalds, le créateur de Linux. Il permet de garder en mémoire toutes les versions d'un fichier. Évidemment ce système de versioning ne se limite pas à l'informatique. Par exemple, cet article est disponible sur Github, le site de partage utilisant le système Git (https://github.com/claudebueno/Document-Git).

Un dépôt Git s'articule sur une branche « master » qui est en quelque sorte la colonne vertébrale du projet. Puis, pour des raisons que j'aborderai plus bas, peuvent se rattacher d'autres branches pour répondre à un besoin temporaire ou pour ajouter des fonctionnalités au projet.

Pour y voir plus clair, je vais aborder quelques questions :

Qu'est-ce qu'un commit ?

Au fur et à mesure que le projet évolue, tu va pousser les mises à jour des fichiers vers le « dépôt » Git. C'est ce qu'on appelle faire un commit.

Le commit est une étape dans l'histoire du projet. Il se matérialise par un titre et une description de la mise à jour. Ainsi, il te sera plus facile de revenir en arrière ou de consulter une version antérieure de ton code (où celui d'un autre développeur si tu travailles sur un projet partagé).

À quoi sert la commande git log ?

A force de faire des commits, tu peux avoir besoin de retracer les différents évènements du projet. C'est à ce moment-là que tu vas faire appel au git log.

Comme son nom l'indique, le git log permet d'obtenir les informations des différents commit de ton projet. D'où l'importance de nommer et commenter chaque commit de manière claire.

Qu'est-ce qu'une branche ?

Au début du projet, tu utilises la branche par défaut, également appelé « master », pour créer un dépôt.

Lorsque le projet commencera à prendre de l'ampleur, Tu seras amené à créer de nouvelles branches pour développer une fonctionnalité particulière sans interférer sur la branche master. Ensuite, tu pourras fusionner cette branche temporaire avec la branche principale.

L'utilisation de branches seras très utilises si tu veux conserver la trace des différentes versions du projet et des fonctionnalités.

Cet article a été rédigé dans le cadre d'un atelier pour la formation « Gérer son code avec Git et GitHub » d'OpenClassrooms : https://openclassrooms.com/courses/gerer-son-code-avec-git-et-github

Liens : https://github.com/