GitKraken, petit guide

Explications, commandes de bases et scénarios courants

10 janvier 2018

1 Description

GitKraken est un utilitaire graphique de git, un logiciel de gestion de version de code. Il est possible de lier un code local avec un code stocké en ligne (github dans notre cas) mais ce n'est ni automatique ni obligatoire. C'est ce que l'on appelle une *remote*, qu'il faut initialisée.

Afin de créer un projet, il faut initialiser un versionning.

2 Commande de base

Afin de créer un projet, il faut faire des "photos", des versions de l'état du projet à différents moments que l'on choisit en fonction de l'avancement. Quand, au cours du développement, on considère que l'état actuel est intéressant, que l'on pourrait avoir envie d'y retourner, avant de testé quelque choses de nouveau, ... on effectue un **commit**. Il comprend un titre et une description et le projet à l'état de la création de celui-ci. Les différents **commits** sont organisés dans des **branchs**. Celles-ci permettent d'avoir en parallèle différentes versions du code. Il est possible de mettre en ligne avec la commande **push** la version locale d'une branche. De la même façon, il est possible de récupérer l'état d'une branche depuis internet sur sa version locale avec la commande **pull**. On peut regrouper deux branches grâce à la commande **merge**. Pour cela il faut faire glisser la branche cible et la lâcher sur la branche destination. Il faut ensuite choisir *Merge NomBranchCible into*[1] *Merge NomBranchDestination* dans l'onglet déroulant qui apparait . Parfois, la gestion des conflits nécessite un choix qui sera présenté avec un face à face des deux codes. On doit choisir la version a conserver, par bloc de texte modifié ou par ligne.

L'utilisation de branches est très pratique et quasiment indispensable pour travailler à plusieurs sur un même projet. Pour cela, à partir d'un état d'un projet, on crée une branche par fonction et sur celle-ci chaque personne qui intervient crée une branche. Chacun effectue ses modifications et ses **commits** sur sa branche et la réuni avec develop quand il le souhaite (le travail est fini, on a besoin d'une version plus récente pour continuer, ...) grâce à la fonction **merge**. Les branches qui n'ont pas besoin d'être partagées n'ont pas forcément besoin d'être mises en ligne (pas besoin de *push*).

1. À ne pas confondre avec onto.

Lorsque l'on travaille à plusieurs sur git (donc avec une version en ligne), il est important de respecter la même organisation :

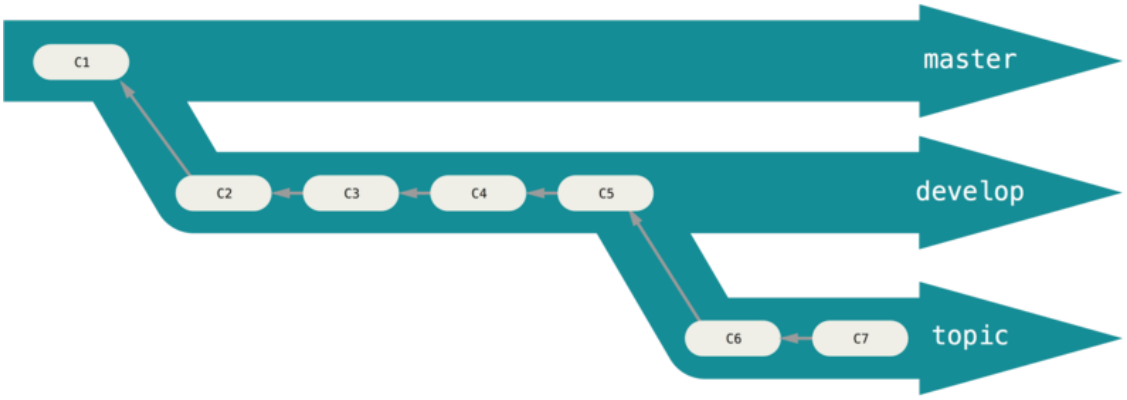

FIGURE 1 – WorkFlow d'un projet git

Sur la figure 1, la branche **master** est la branche principale où il ne doit y avoir que des versions fonctionnelles du code. A partir de celle-ci, est créer une branche **develop** où seront fait tous les sous-branchements nécessaires pour faire une nouvelle version fonctionnelle. C'est depuis cette banche que l'on va faire une branche par fonction (**topics**). Il est nécessaire, avant chaque **merge**, de **pull** la branche destination afin d'ajouter nos modifications sur une version à jour de la branche de destination.