1 – Qu'est ce qu'un commit ?

Avant de coder, on prend souvent quelques instants de réflexion pour se fixer un but, pour s'éclaircir les idées. Après cette étape, on se lance dans le code. Au fur et à mesure qu'on avance, notre fichier de code source qui était vide se remplit. A un moment donné, on souhaite faire des tests pour voir si le code fonctionne comme prévu. Ok supposons que ton premier test s'appelle test 1 et le code actuel, code 1. Après avoir testé code 1, tu obtiens résultat 1. Etant donné qu'on essaie d'atteindre la perfection d'autres tests se succèdent par la suite, s'accompagnant d'une modification de code ou non à chaque fois.

Voici ce qui se passe :

CODE 1 / TEST 1 / RESULTAT 1

CODE 2 / TEST 2/ RESULTAT 2

CODE 3 / TEST 3/ RESULTAT 3
.
.
.
.
CODE n / TEST n / RESULTAT n

A un moment, tu mets fin aux tests et tu te dis CODE 15 / TEST 15 / RESULTAT 15 me satisfait, tu penses que c'est le meilleur. Mais il y'a un problème tu es à ce niveau CODE 40 / TEST 40 / RESULTAT 40. Tu te dis, ok je vais modifier CODE 40 pour obtenir de nouveau CODE 15. De toi à moi, on sait que tout se joue dans les détails. Tu peux retrouver ton CODE 15 mais après combien de temps ? T'es peut être fatigué(e), auras tu la force de reproduire fidèlement CODE 15 ?

Tu te demandes , pourrai je faire mieux ? Moi tout souriant, je te dis OUI :) . En utilisant GIT, tu peux sauvegarder chaque modification qui t'intéresse. Dans notre cas, tu peux sauvegarder tes 40 CODES. Après, il te suffira de revenir sur tes pas et de sélectionner puis télécharger CODE 15. T'auras plus à réfléchir sur les détails du genre à CODE 15, j'avais utilisé un for ou un while ? Est ce que j'avais utilisé une variable globale ou locale ? La liste est longue ... Chaque modification que tu enregistres, constitue un COMMIT :) !

2 – A quoi sert la commande git log ?

Plus haut, je t'ai dit que tu peux revenir sur tes pas. Concrètement ça veut dire que tu as accès à l'historique de tes COMMITS. Tu vois où je veux en venir ? :) Tu arrives à faire le lien ? Peu importe, GIT LOG te permet d' avoir accès à l'historique de tes COMMITS :) .

3 – Qu'est ce qu'une branche ?

T'aimes monter sur des arbres ? Tu te demandes sûrement où est le rapport avec GIT ? Ok t'as raison. A première vue, il n y a pas de rapport. Mais entrons un peu en profondeur. A la partie 1, on a été d'accord qu'on se fixe un but précis avant de coder. Pour atteindre ce but, tu as à ta disposition plusieurs moyens. De même, pour atteindre un niveau d'un arbre, tu as le choix d'utiliser une multitude de branches.

Dans le contexte de GIT, une branche est un concept qui te permet de te focaliser sur le but que tu t'es fixé. Ainsi, en créant une branche (analogie avec les arbres : en choisissant d'utiliser une branche), tu écriras du code ou du texte par rapport à ce que tu vises. Bien sûr que tu peux faire autant de commits que tu désires et en utilisant git log, t'auras accès aux modifications liées à la branche sur laquelle tu te trouves.

OK, je reconnais que c'est un peu abstrait mais avec de la pratique tu y parviendras.