Git est un outil de suivi de versions, qui permet de "suivre" les modifications apportées aux fichiers indexés dans un projet.

I. Qu'est-ce qu'un commit.

La définition dans la littérature d'un commit est la suivante. Un commit est un état précis des fichiers du projet indexés avec git. Cet état est identifié par une chaine de caractères réputée unique (un hash).

Voyons ce qu'est un commit en expliquant la séquence qui permet de créer un nouveau commit :

1. Les modifications apportées aux fichiers sont listées dans le projet, et peuvent être affichées par la commande qui affiche l'état actuel du répertoire de travail du projet par rapport à l'index de « git » : git status

   Cela signifie à ce stade que ces modifications existent dans l'état actuel du projet mais qu'elles n'ont pas encore été "validées" pour le prochain "commit".

2. Ces modifications doivent alors être "validées", c'est-à-dire qu'il convient d'indiquer lesquelles doivent être inclues dans ce prochain "commit", ce que l'on fait avec la commande suivante qui ajoute tous les fichiers déjà suivis à l'index (mais n'ajoutera pas les fichiers que l'on vient de créer) : git add .

3. Le ou les fichiers modifiés (et déjà indexés) sont alors "prêts à être commités", c'est-à-dire prêts à être inclus dans le prochain commit. Une variante de cette commande permet d'ajouter de nouveaux fichiers à l'index (ou de prendre en compte les fichiers supprimés) :git add -A.

4. Le « commit » lui-même est créé par la commande suivante, qui va créer un nouveau commit, et ajouter un commentaire pour décrire la raison et/ou le contenu des modifications : git commit -m "Raison du commit"

Un commit peut être vu comme une étape de plus dans l'avancement d'un projet.

Un commit peut aussi bien consister en la modification d'un seul caractère, qu'en l'ajout d'une librairie complète au projet., cela ne dépend que de la manière de les utiliser.

II. À quoi sert la commande git log.

La commande « git log », invoqué telle quelle sans argument, va énumérer en ordre chronologique inversé les commits réalisés. Les plus récents apparaissent en premier, les plus anciens en dernier.

Chaque invocation de cette commande donne beaucoup d'autres informations, telles que :

- Sa somme de contrôle SHA.
- Le nom de l'auteur.
- L'e-mail de l'auteur.
- La date du commit.
- Le message du commit.

III. Qu'est-ce qu'une branche.

Une branche est une dérivation à partir du "tronc commun" d'un projet, avec lequel elle peut ensuite être fusionnée.

C'est une manière de cloisonner certains développements, comme par exemple le développement d'une nouvelle fonctionnalité ou d'un patch, mais aussi de maintenir et faire évoluer en parallèle plusieurs versions d'un logiciel.

Il est possible de passer d'une branche à l'autre afin de travailler successivement dans plusieurs parties d'un projet.

Une branche peut être abandonnée, ou fusionnée avec une autre branche. Dans ce cas, il peut être nécessaire de gérer des conflits de fusion, lorsque certains fichiers ont évolué différemment dans les deux branches.

Le processus de création, travail et fusion d'une branche peut être résumé avec les commande suivantes :

1. Créer une nouvelle branche et se positionner dessus : git checkout -b ma_branche. On travaille ensuite normalement dans "ma_branche", les commits étant dès lors enregistrés dans cette branche.

2. Pour changer de branche, on utilise : git checkout autre_branche.

3. Une fois le travail dans une branche terminé, on se repositionne sur la branche d'origine pour y intégrer son travail : git checkout master puis git merge ma_branche.