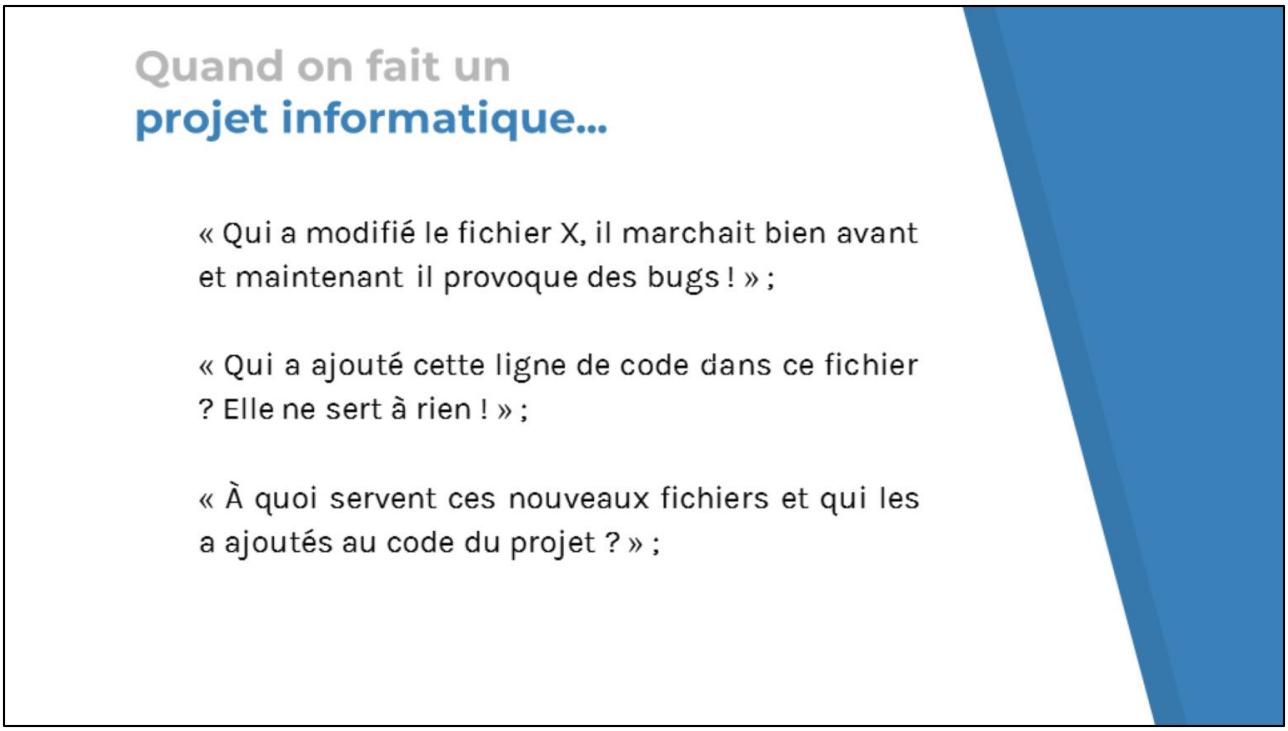
Gestion des versions de code avec Git

Charles Tholliez
Simon Hay

Formation de formateurs
M2 GPhy - 2017

1.

Logiciel de gestion de version : **Kézako ?**



Pour éviter ce genre de problèmes lors d'un projet informatique → Logiciel de gestion de version

Avantages et objectifs
d'un logiciel de gestion de version

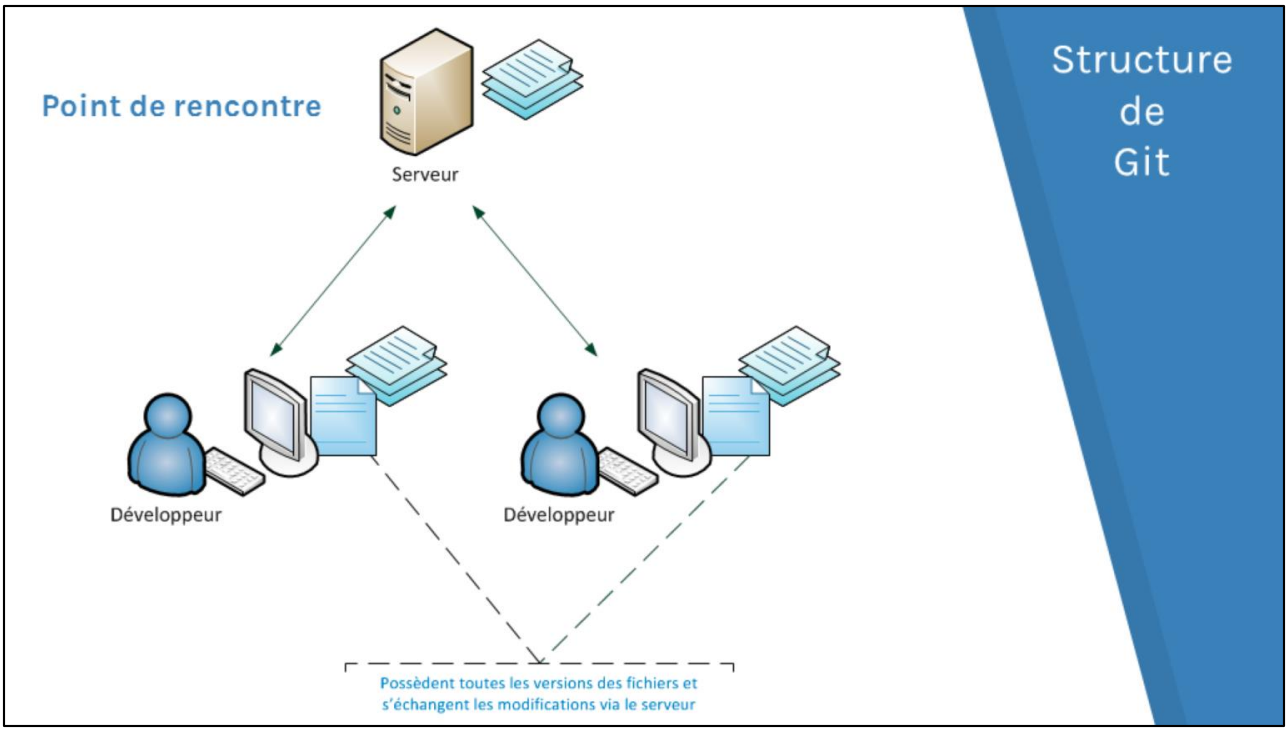
résolution des problèmes listés précédemment ;
suivre l'évolution d'un code source : Git est capable de dire qui a écrit chaque ligne de code de chaque fichier et dans quel but ;
travailler à plusieurs : Git est capable d'assembler (de fusionner) leurs modifications et d'éviter que le travail d'une de ces personnes ne soit écrasé.

Différents logiciels de gestion de version, présentation de Git dans ce cours

Différents
logiciels de gestion de versions

SUBVERSION®

2016 :
12 M d'utilisateurs
(#1 dans le monde)



Le serveur sert de point de rencontre entre les développeurs.
Ils s'échangent les modifications via le serveur.
Ce dernier possède également les fichiers avec les dernières modifications.

- Git est à l'origine prévu pour Linux. Il existe des versions pour Windows mais pas vraiment d'interface graphique simplifiée. Il est donc à réserver aux développeurs ayant un minimum d'expérience et… travaillant de préférence sous Linux.
- Git est lié à des sites web collaboratifs comme GitHub. GitHub peut être comparé à un réseau social pour développeurs. Sur github tout le code est public. Il est possible de le rendre privé mais c'est payant. On peut donc se mettre sur un projet et collaborer avec d'autres gens si l'on veut. Il s'agit donc d'un logiciel pour gérer un dépôt git.

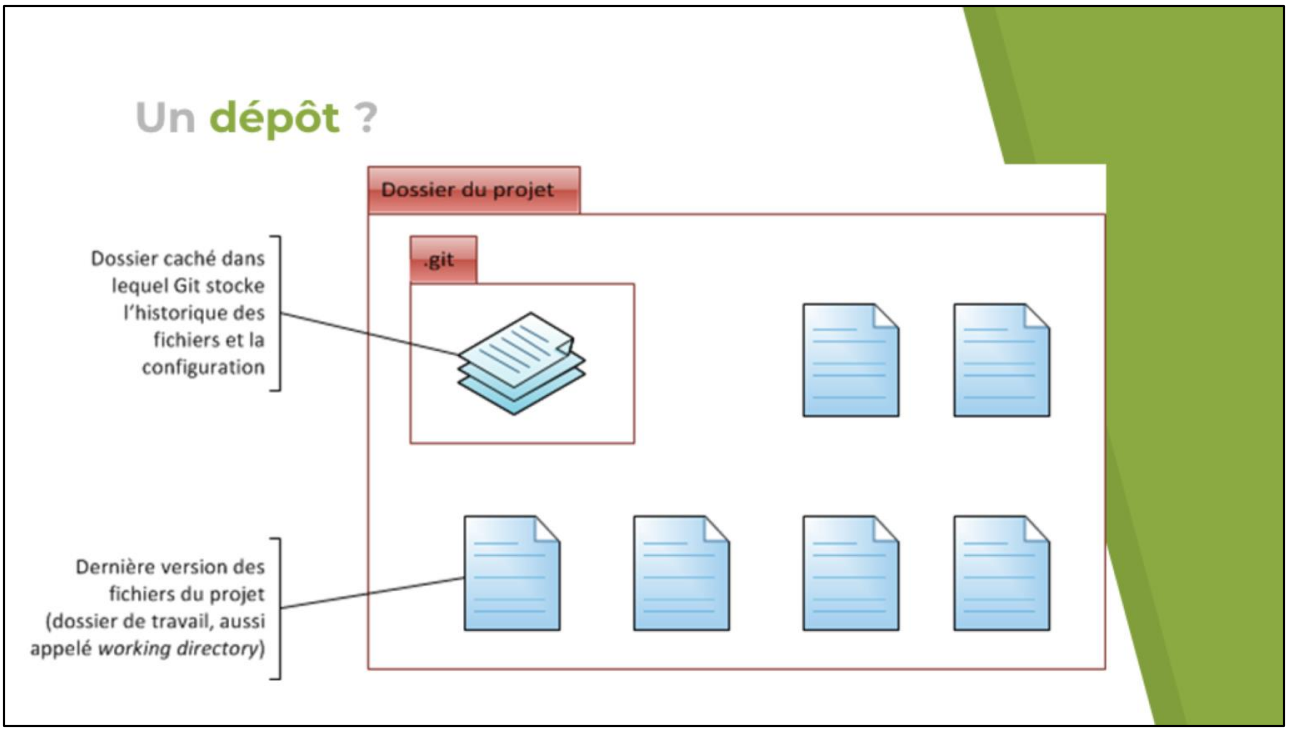
2.

Commencer à travailler avec **Git**

Deux solutions...

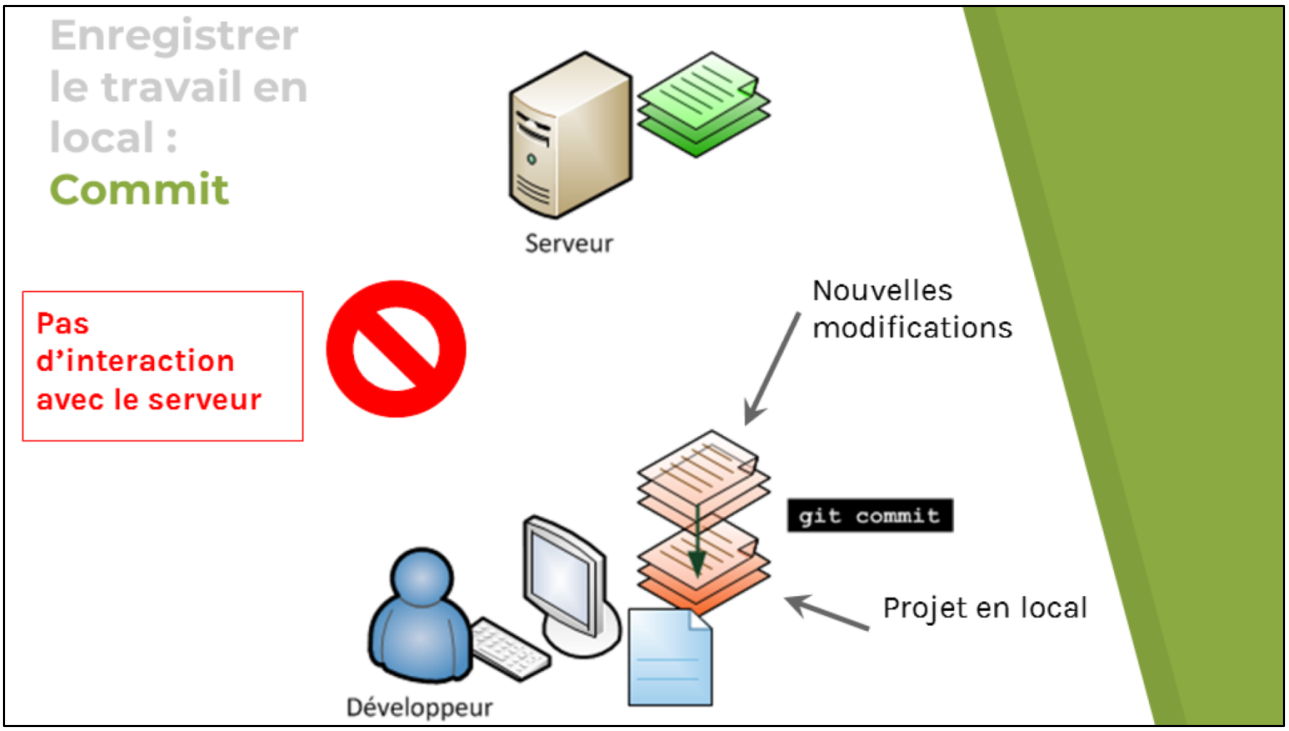
→ création d'un dépôt vide pour commencer un nouveau projet.

→ clone d'un dépôt existant, c'est à dire récupération de tout l'historique des changements d'un projet pour pouvoir travailler dessus.



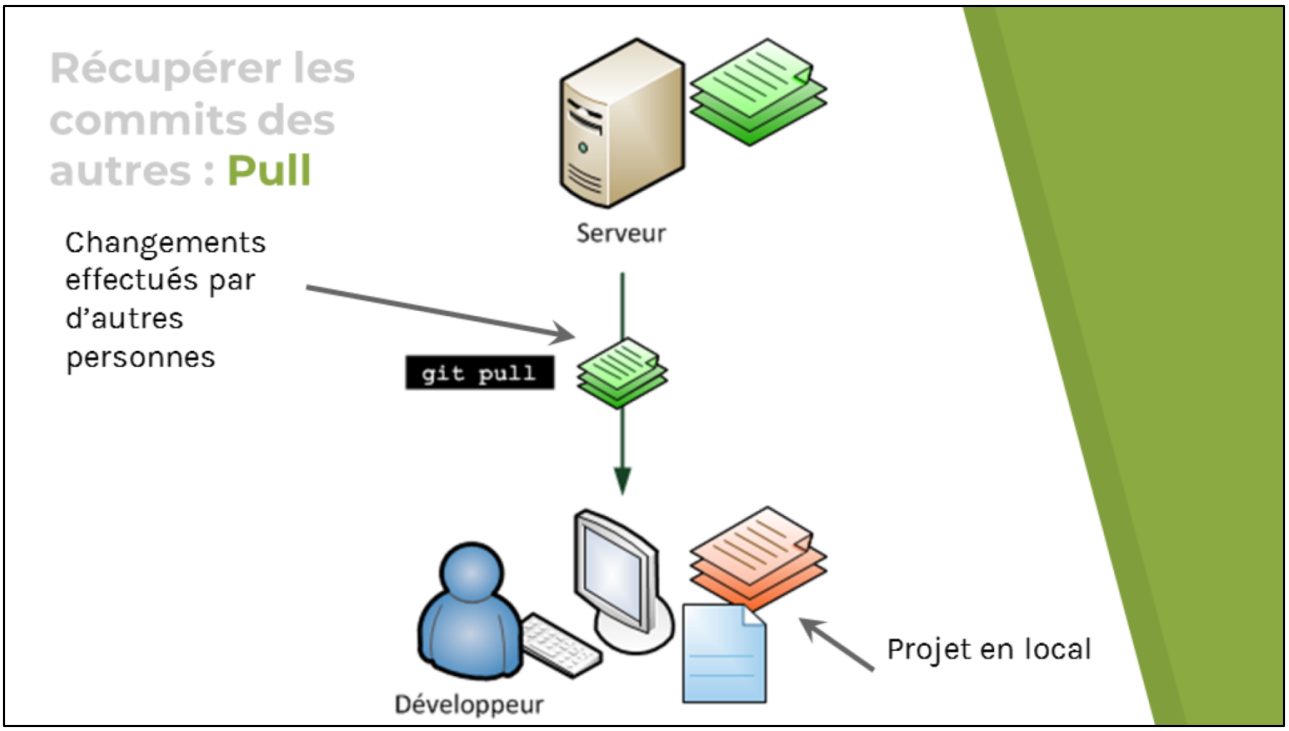
Dans un logiciel de gestion de versions comme Git, un dépôt représente une copie du projet. Chaque ordinateur d'un développeur qui travaille sur le projet possède donc une copie du dépôt. Dans chaque dépôt, on trouve les fichiers du projet ainsi que leur historique.

- Au début, Git crée tout simplement un dossier caché .git à la racine du dossier de votre projet. Il contient toute la configuration du dépôt Git et toutes les informations concernant son historique.
- À part ce dossier .git, il y a dans votre dossier de projet tous les fichiers le composant. C'est ces fichiers là qui seront modifiés.



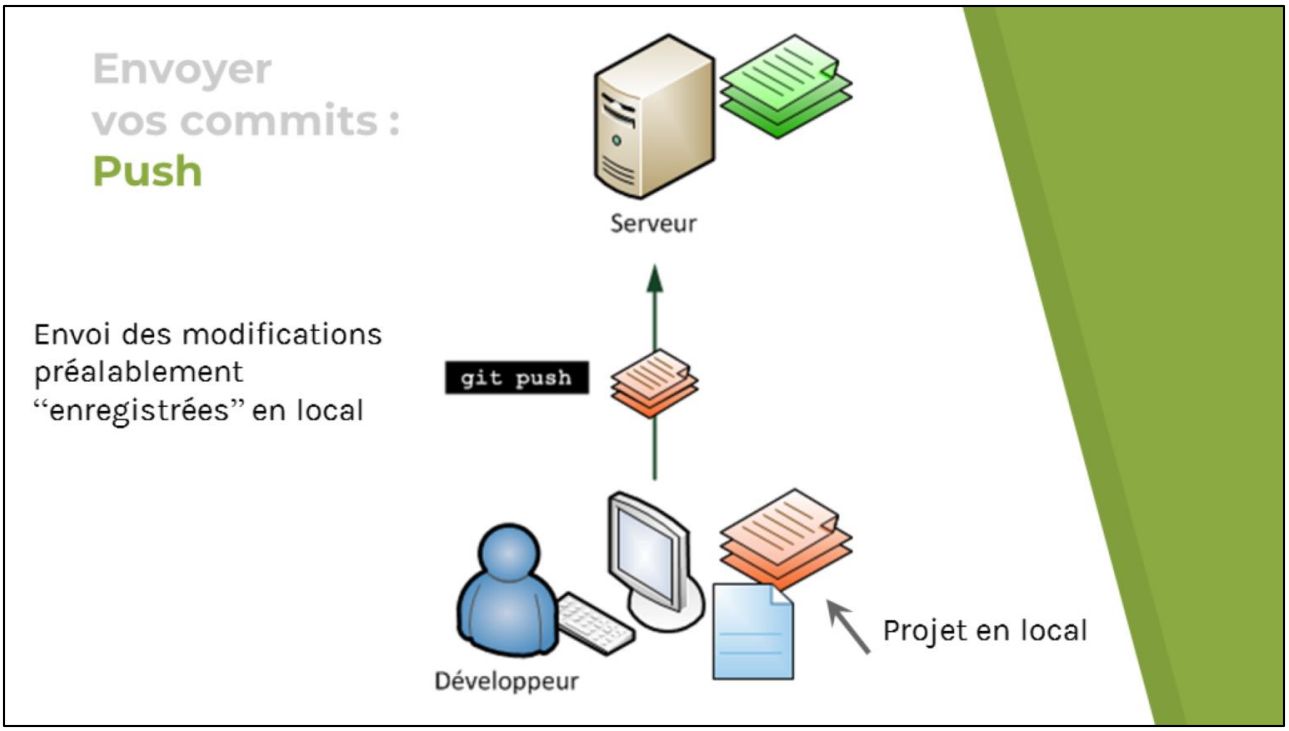
Lorsqu'un commit est effectué, les modifications sont enregistrées dans l'historique des modifications, comme l'illustre le schéma ci-dessous. Avec Git, un commit est local, c'est à dire que personne ne sait que le commit a été effectué, à part vous (en cas d'erreur, il y a la possibilité de l'annuler). Plus tard, nous verrons comment envoyer ce commit sur le serveur, afin que tout le monde puisse voir vos modifications.

Il est très important de mettre un message clair et précis sur ce qui a été modifié.



Pour télécharger les modifications apportés par les autres membres du projet, il faut faire ce que l'on appelle un "pull". Cela récupère les nouvelles informations stockées sur le serveur réalisées par d'autres personnes et les intègre sur votre dépôt local.

- soit vous n'avez effectué aucune modification depuis le dernier pull, dans ce cas la mise à jour est simple (on parle de mise à jour fast-forward) ;
- soit vous avez fait des commits en même temps que d'autres personnes. Les changements qu'ils ont effectués sont alors fusionnés aux vôtres automatiquement.



Pour envoyer les modifications sur le serveur après les avoir “enregistrées” en local avec un commit, il faut faire un “push”.
Le changement vers le serveur doit être de type fast-forward car le serveur ne peut régler les conflits à votre place s’il y en a. Personne ne doit avoir fait un push avant vous depuis votre dernier pull.

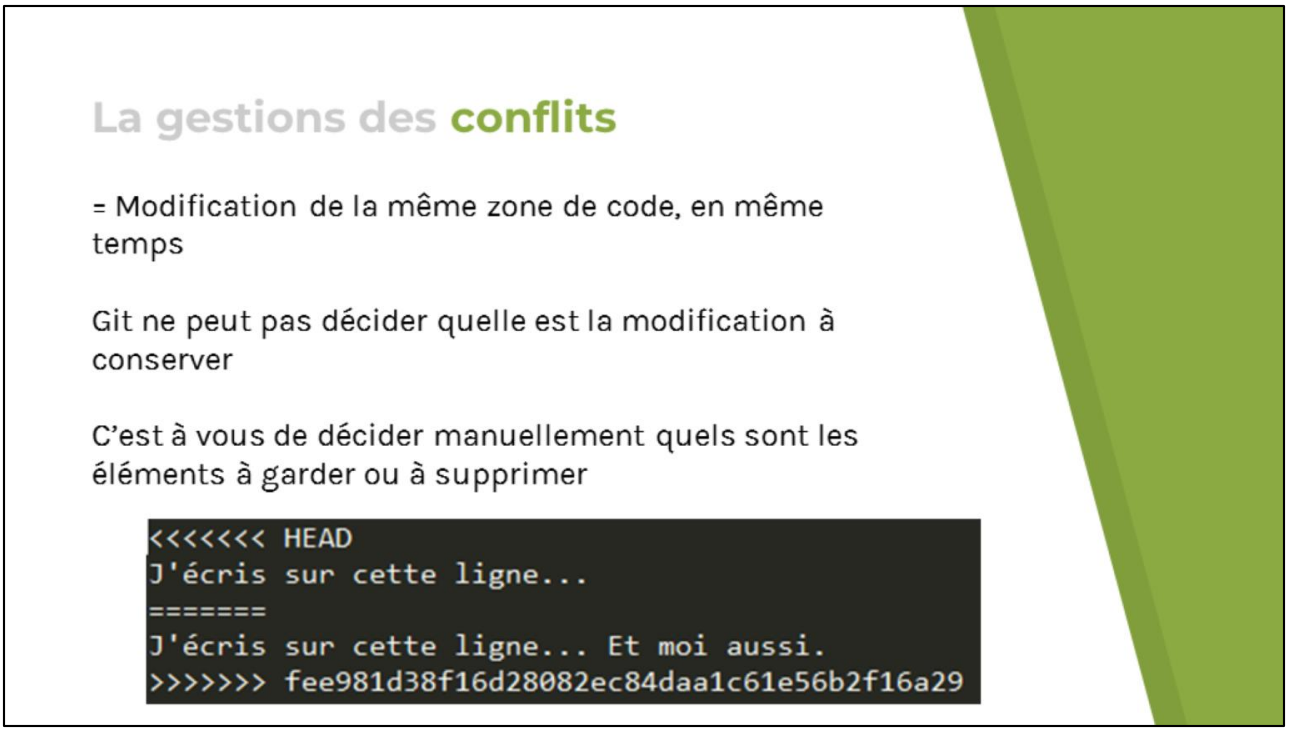

I WANT YOU TO
COMMIT PULL PUSH !

Annuler proprement un commit : Revert

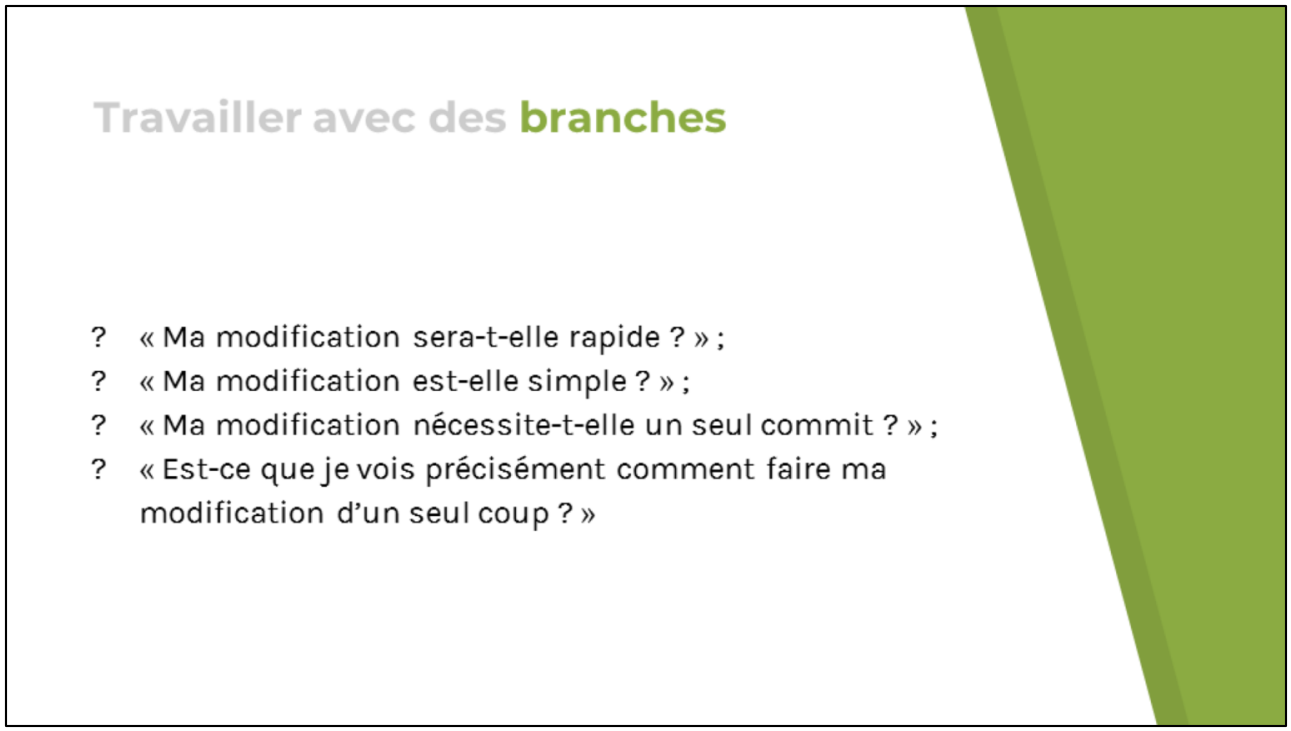
2 cas de figures :

- Commit “non-poussé” (enregistrement local) : reset
- Commit “poussé”, donc enregistré sur le serveur : revert

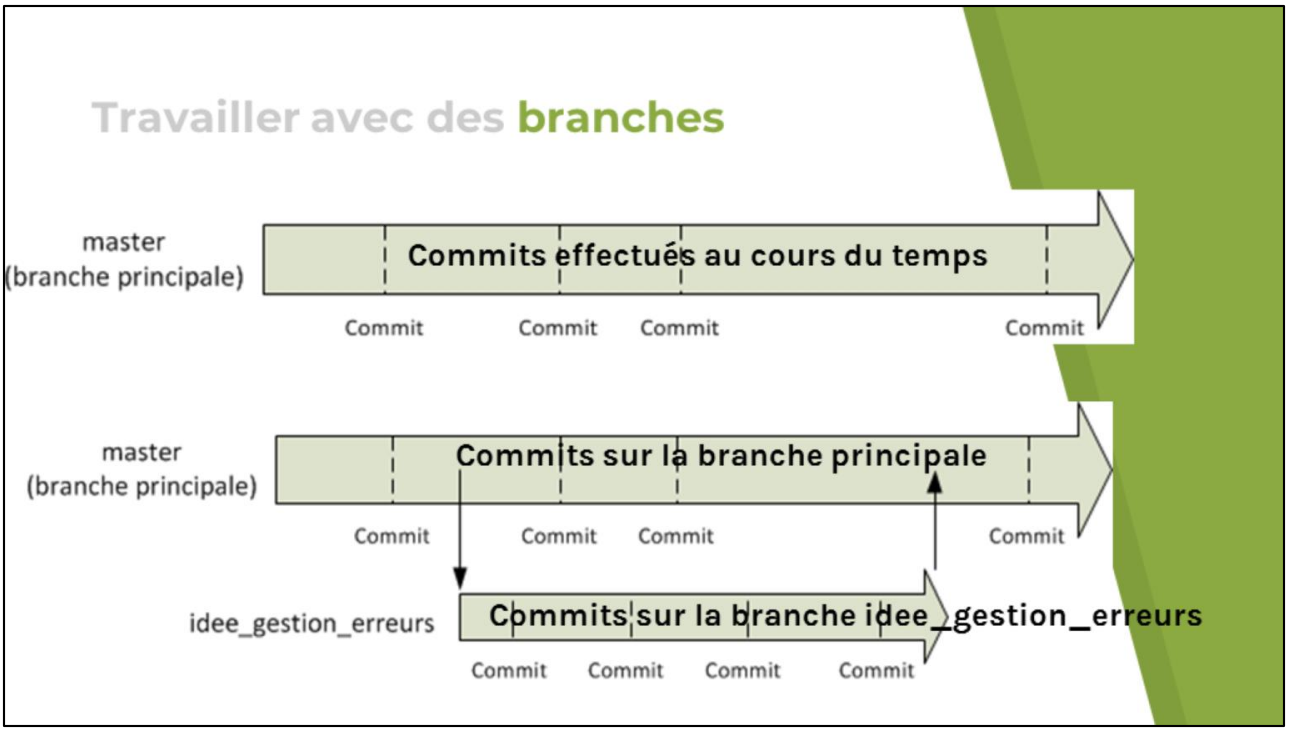
https://makina-corpus.com/blog/metier/archives/git-annuler-proprement-un-commit-apres-un-push



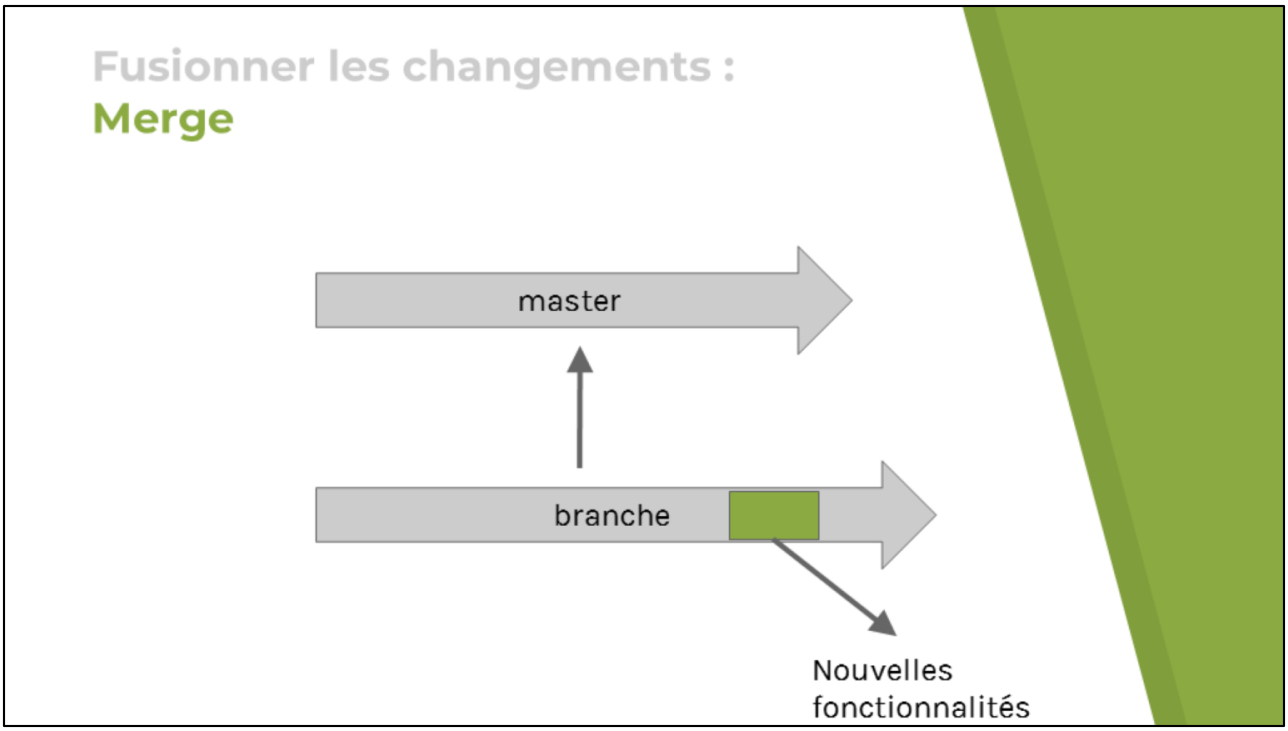
Quand deux personnes modifient la même zone de code en même temps, Git dit qu'il y a un conflit : il ne peut décider quelle modification doit être conservée et indique alors le nom des fichiers en conflit. Ouvrez-les avec un éditeur et recherchez une ligne contenant « <<<<<<<<< ». Ces symboles délimitent vos changements et ceux des autres personnes. Supprimez ces symboles et gardez uniquement les changements nécessaires, puis faites un nouveau commit pour enregistrer tout cela.



Si la réponse à l'une de ces questions est « non », vous devriez probablement créer une branche. Créer une branche est très simple, très rapide et très efficace. Il ne faut donc pas s'en priver.



- Flèche du haut : Dans Git, toutes les modifications que vous faites au fil du temps sont par défaut considérées comme appartenant à la branche principale appelée « master »
- 2 flèches du bas : Supposons que vous ayez une idée pour améliorer la gestion des erreurs dans votre programme mais que vous ne soyez pas sûrs qu'elle va fonctionner : vous voulez faire des tests, ça va vous prendre du temps, donc vous ne voulez pas que votre projet incorpore ces changements dans l'immédiat.
  Il suffit de créer une branche, que vous nommerez par exemple « idee_gestion_erreurs », dans laquelle vous allez pouvoir travailler en parallèle
  Il est donc possible de travailler sur la nouvelle fonctionnalité sur la branche secondaire tout en continuant de faire des commits sur la branche principale



Vous connaissez déjà le merge : quand on fait un pull, en réalité on fait un fetch (= téléchargement des commits) puis un merge, qui permet de fusionner les commits téléchargés avec le code


BRATISLA
BOYS
STACH STACH
LE TITRE RÉVÉLÉ par le
MORNING LIVE
funradio

Quand vous devez faire un merge ou un checkout (changement de branche) mais qu'il y a des modifications non sauvegardées en cours sur vos fichiers, Git va vous demander d'y remedier. Soit vous commitez les changements, soit vous les annulez, sans quoi pas de merge ni de checkout possible.

Il existe une solution : Git stash.

Au travail !

Charles Tholliez
Simon Hay

Formation de formateurs
M2 GPhy - 2017