
git

Mise en situation

Le projet de développement TRETA est lancé au sein d'une entreprise.

4 développeurs sont affectés à sa réalisation :

- Vincent
- Arthur
- Evelyne
- Victor

Problème : Ils travaillent à distance. Comment vont-ils échanger leurs développement sur le projet ?

Mise en situation

Première idée : l'échange de mail !

Voilà ce que ça peut donner :

Mise en situation

Première idée : l'échange de mail !

Voilà ce que ça peut donner :

- Mauvaise transmission du code
  - Oublie de certaines lignes
  - Mauvais numéro de ligne
- Mise à jour du code après un long moment

Mise en situation

Première idée : l'échange de mail !

Voilà ce que ça peut donner :

- Mauvaise transmission du code
  - Oublie de certaines lignes
  - Mauvais numéro de ligne
- Mise à jour du code après un long moment

- =>Nuits blanches, engueulades...

Mise en situation – Conclusion

<u>Conséquences :</u>

Le client s’est rétracté

Le projet est tombé à l’eau

L’équipe de développement s’est faite sermonner

 Echange par mails ou clés USB sont donc à proscrire !

Nouvelle solution : serveur décentralisé

Pour éviter ce genre de situation il existe des logiciels de gestion de version décentralisés.

Ce sont des logiciels qui reposent sur des serveurs décentralisés :

- Chaque développeur dépose ses modifications sur le serveur, on dit que le développeur « pousse » ses modifications
- Chaque développeur peut récupérer les modifications des autres, tout en gardant ses ajouts personnels. On dit que le développeur « tire » les sources.

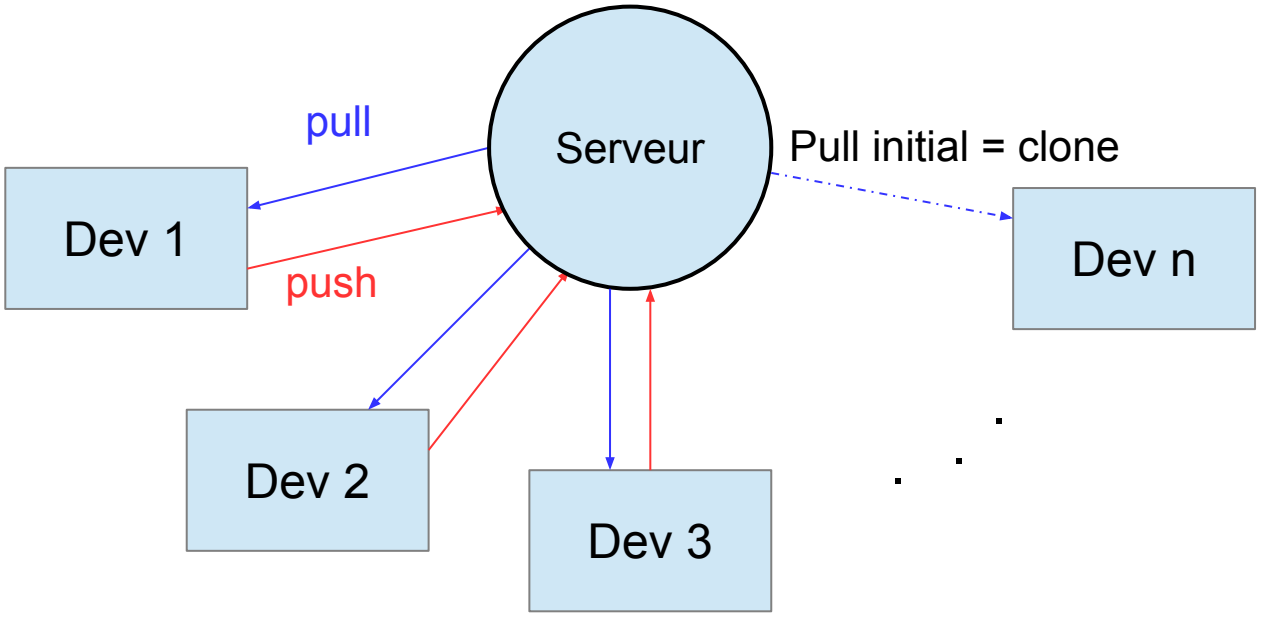
Plusieurs logiciels existent :

- SVN
- Mercurial
- Git
- Etc…

Nouvelle solution : serveur décentralisé

Pour éviter ce genre de situation il existe des logiciels de gestion de version décentralisés.

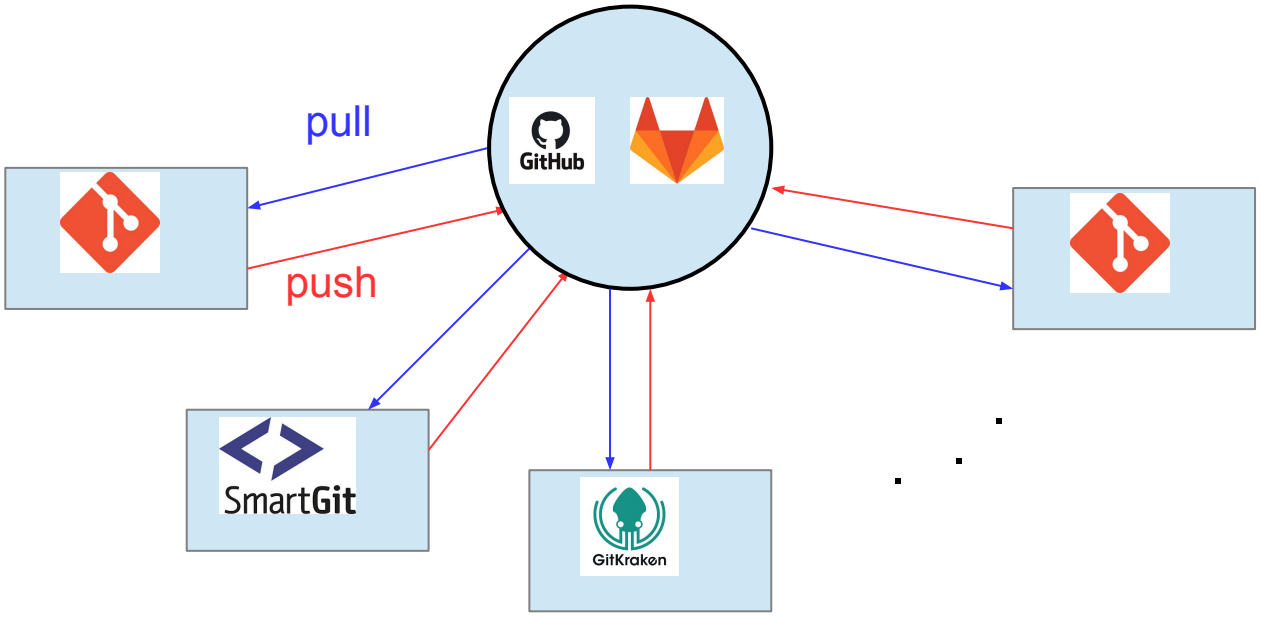
Ce sont des logiciels qui reposent sur des serveurs décentralisés :



Nouvelle solution : serveur décentralisé

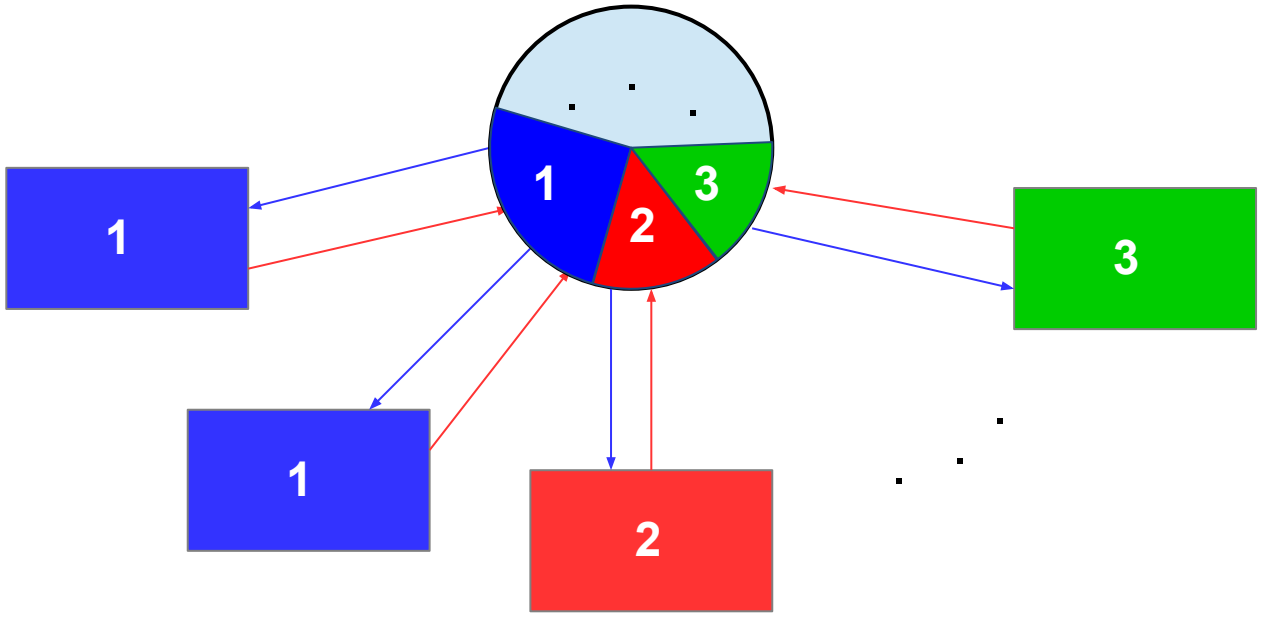
Plusieurs logiciels existent :

- Des clients (git, smartgit, gitKraken...)
- Des serveurs (gitHub, gitLab...)



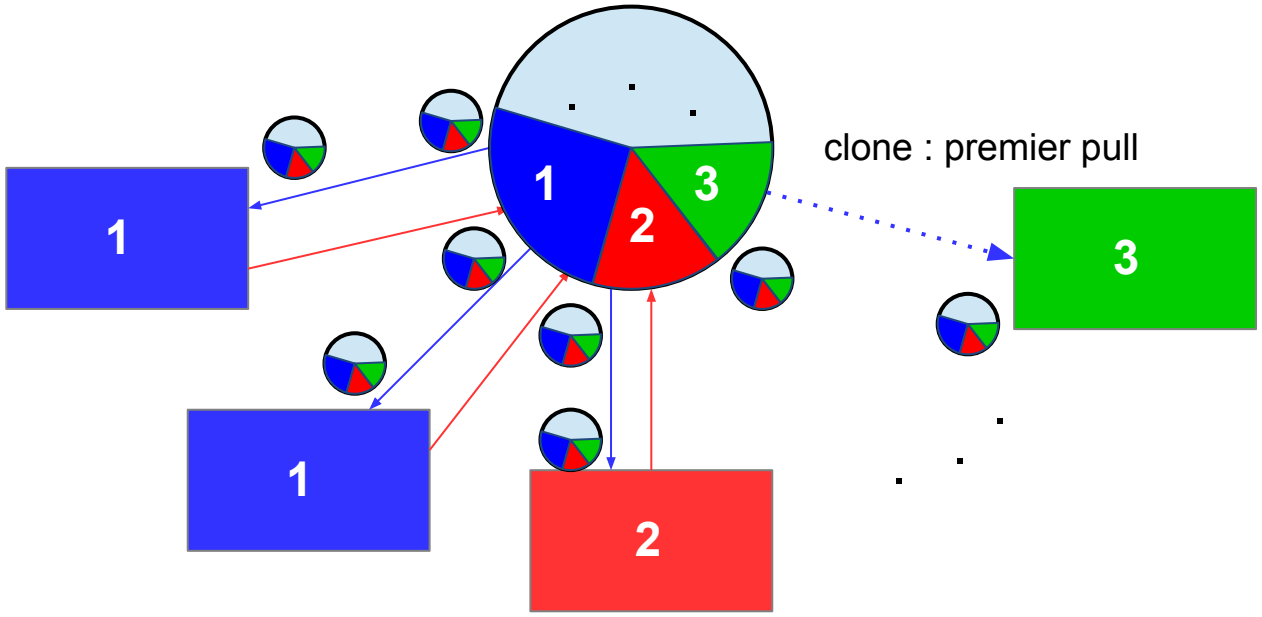
Nouvelle solution : serveur décentralisé

Plusieurs versions (branches) du code cohabitent



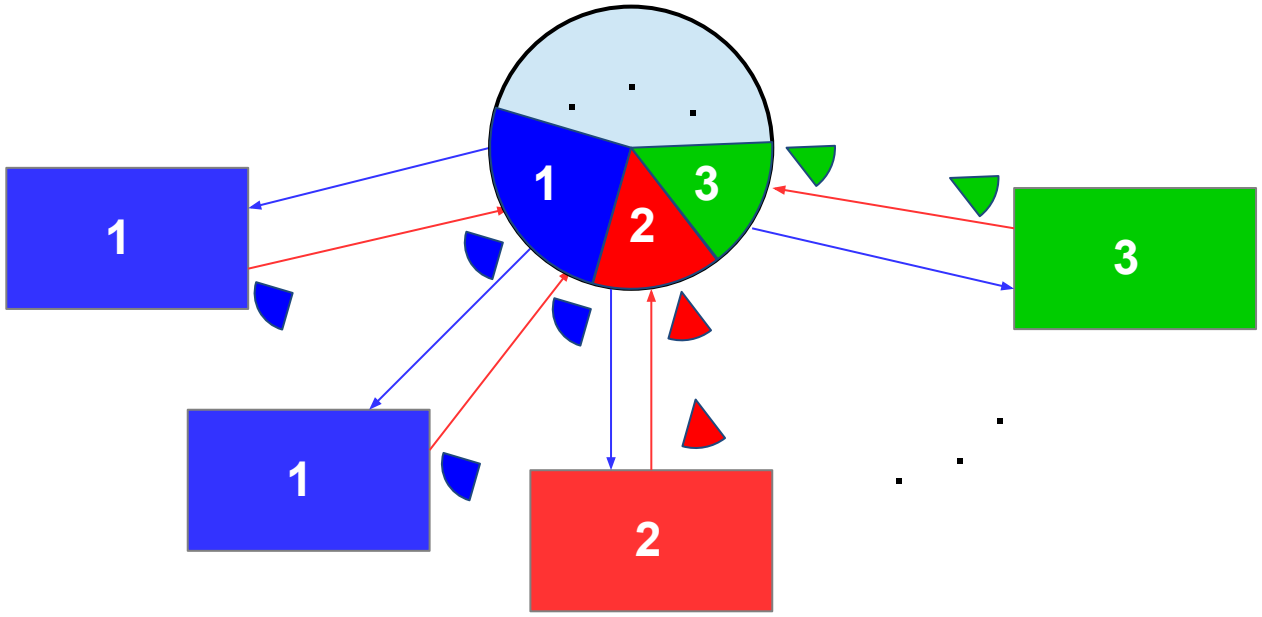
Nouvelle solution : serveur décentralisé

On pull toute les versions

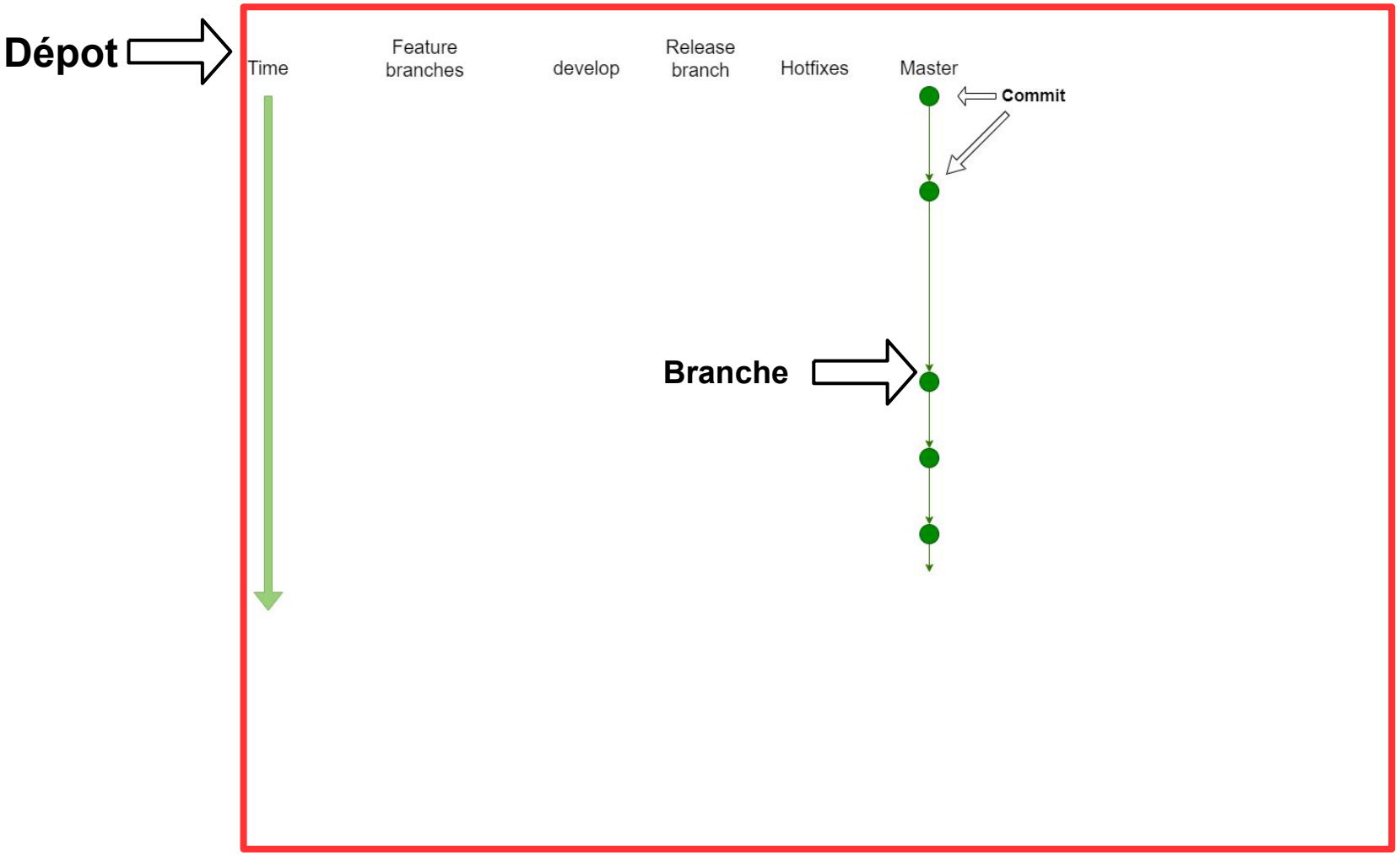


Nouvelle solution : serveur décentralisé

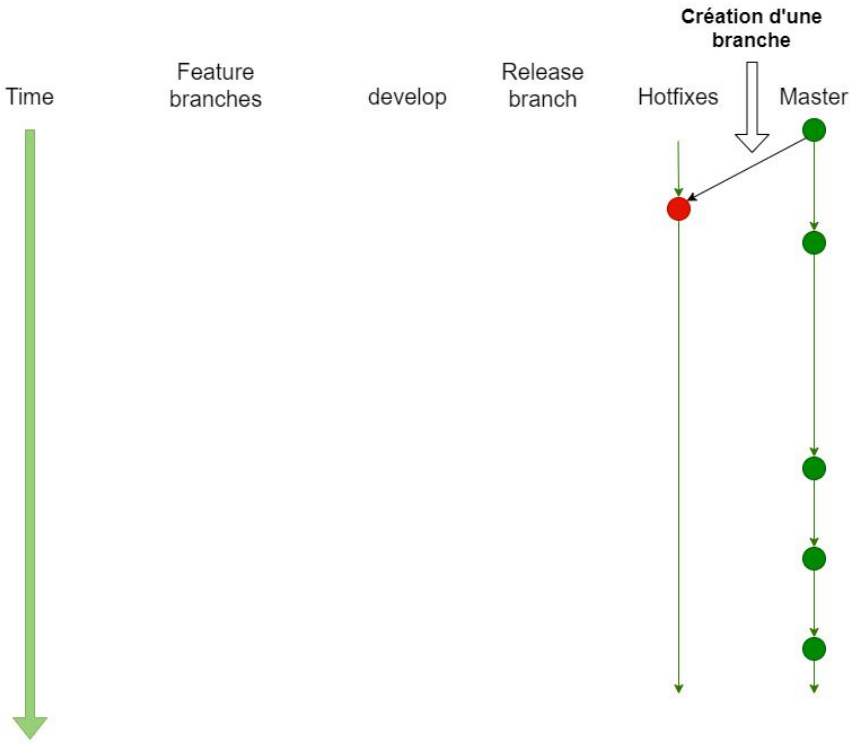
On push sa version



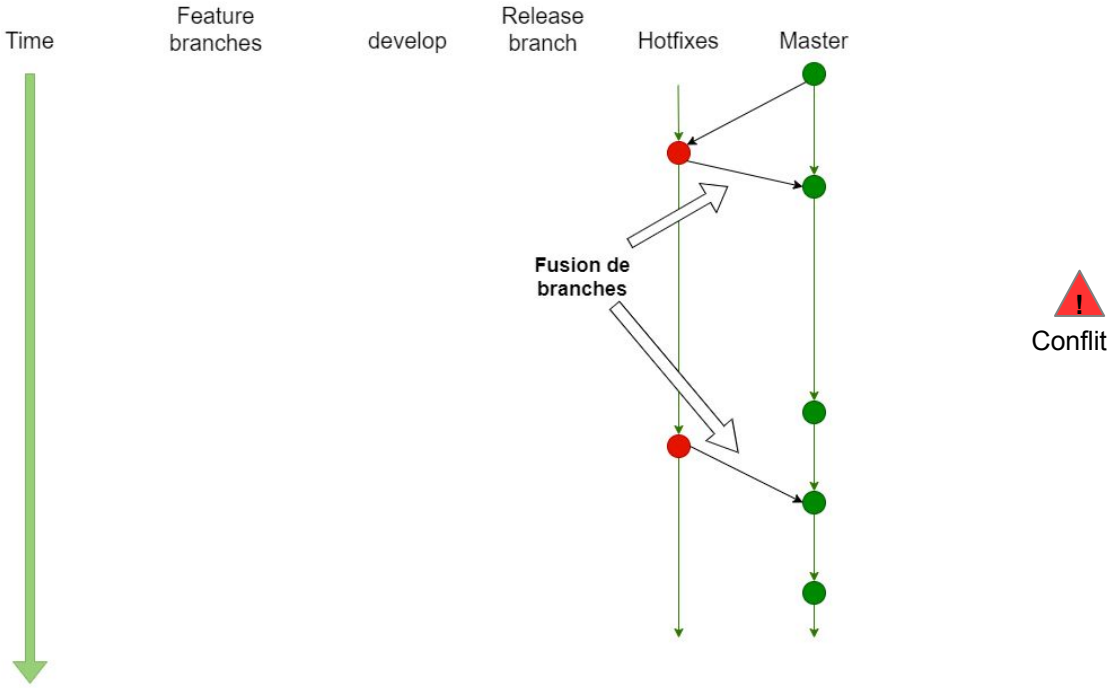
Arborescence git & Vocabulaire



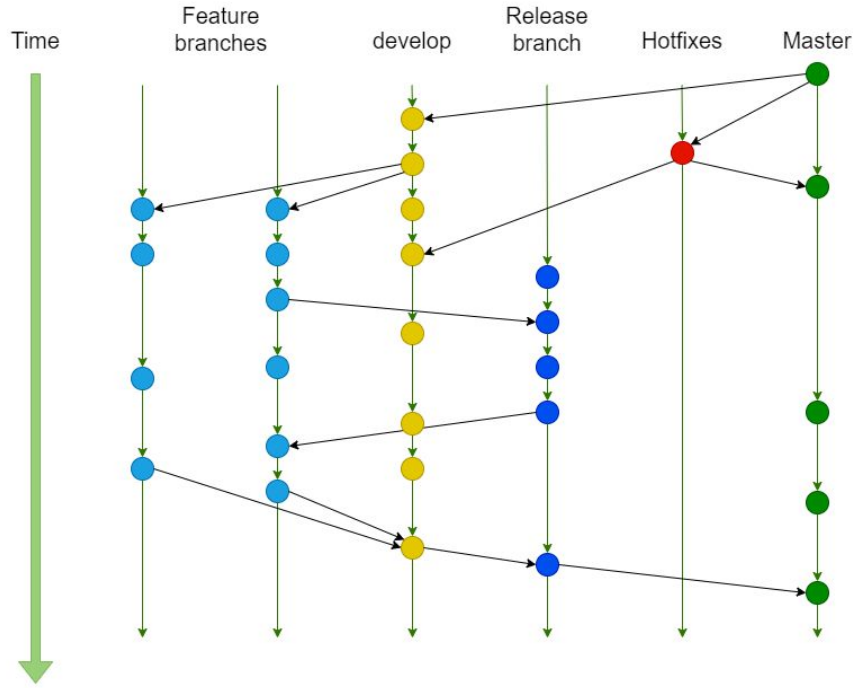
Arborescence git



Arborescence git



Arborescence git



Un peu de vocabulaire

Le fork

- Lorsque vous vous êtes créé un compte sur GitLab, vous avez un espace personnel dans lequel vous pouvez créer vos propres projets, mais aussi de dupliquer un projet existant venant d'un espace d'un autre utilisateur. Dans le cas de la duplication, on parle de fork.
- Le fork se fait depuis la plateforme GitLab.
- Les développeurs pourront alors travailler soit sur le dépôt d'origine, soit sur le dépôt dupliqué. Attention, le dépôt dupliquer devient un nouveau projet à part entière.

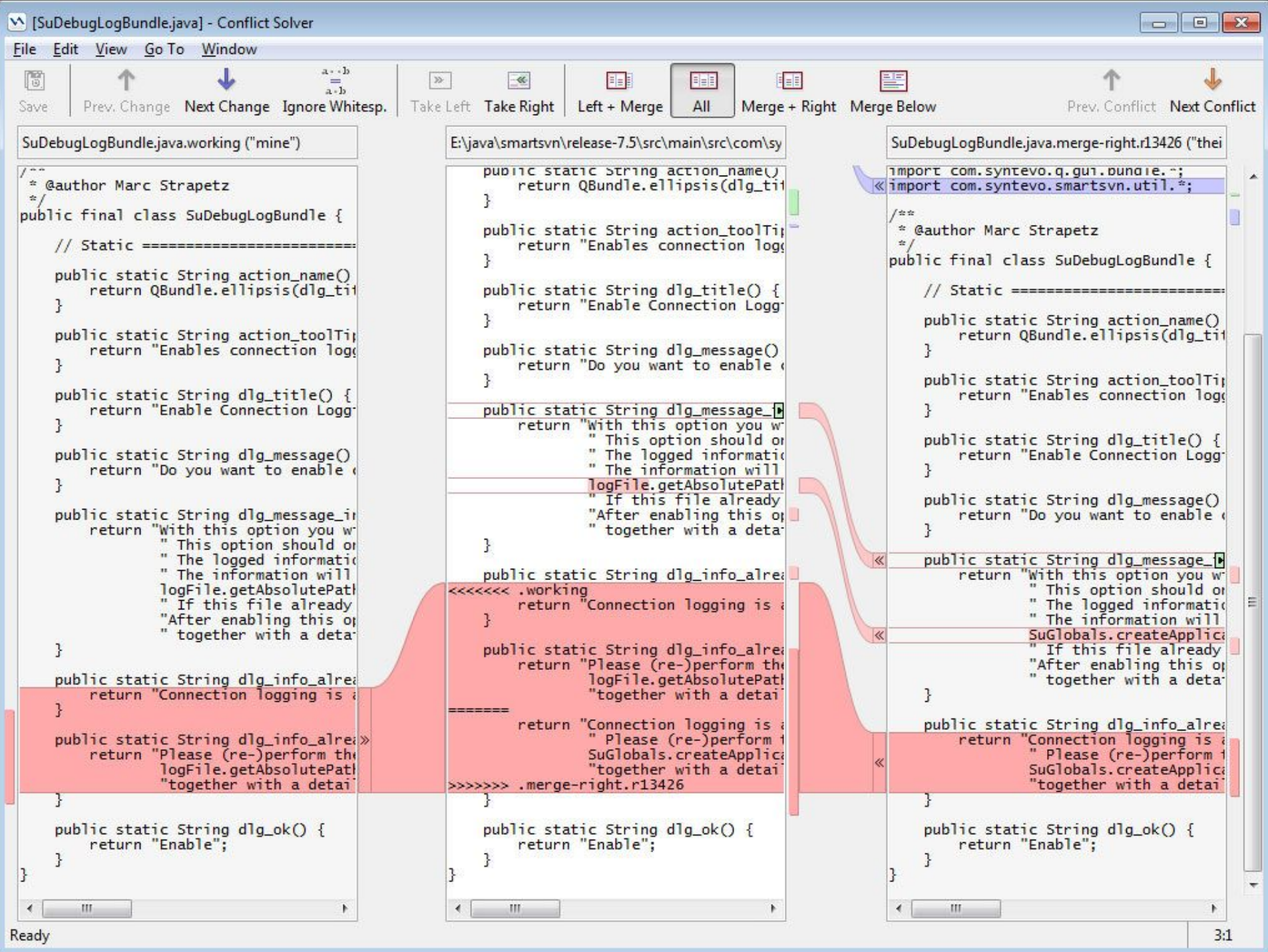
Le Clone

- Cloner un dépôt revient à copier ce dépôt sur notre ordinateur en local.
- C'est à partir de ce clone que nous allons pouvoir créer de nouvelle branche et développer.

Un peu de vocabulaire

Gestion des conflits

- Il est tout a fait possible que plusieurs développeurs travaillent sur les mêmes fichiers, voire les mêmes lignes de ces fichiers.
- Lorsque l'on veut tirer les source ou merger des branches, il risque donc d'y avoir des conflits. Git le signalera et il faudra alors les gérer à la main : c'est-à-dire garder nos modifications et celles des autres développeurs !
- Ce genre d'opération est délicate, et c'est souvent à ce moment là qu'on fait n'importe quoi !
- Il est possible d'utiliser un éditeur de texte pour gérer les conflits, mais il est encore mieux d'utiliser un outil adapté (comme smartgit, voir la fiche donnée sur ce logiciel)


[SuDebugLogBundle.java] - Conflict Solver
File Edit View Go To Window
Save
Prev. Change
Next Change
Ignore Whitesp.
Take Left
Take Right
Left + Merge
All
Merge + Right
Merge Below
Prev. Conflict
Next Conflict
SuDebugLogBundle.java.working ("mine")
E:\java\smartsvn\release-7.5\src\main\src\com\sy
SuDebugLogBundle.java.merge-right.r13426 ("thei
<<<<<<< .working
=======
>>>>>>> .merge-right.r13426
Ready
3:1

Un peu de vocabulaire

J’ai poussé une modification foireuse… Comment faire ?

- Pas de panique, il est possible de revenir en arrière avec un git revert.
- Attention, un git revert va effacer le dernier commit et toutes les modifications qui vont avec, votre code sera donc perdu et à refaire.
- Un conseil : ne pousser jamais de modifications foireuses sur le serveur ! Pensez toujours à tester le code avant ! ;)

Quelques commandes de base

Tout peut se faire en ligne de commande, mais tout peut aussi se faire avec des logiciels comme smartgit.

Dans ce cours nous verrons les lignes de commande ! Il nous faudra donc un terminal (et un terminal Linux de préférence car Git est natif sous Linux, et non sous Windows).

Cloner un dépôt

- Depuis un terminal, placez-vous dans le répertoire où vous souhaitez cloner le dépôt distant (rappel : pour se positionner dans un répertoire depuis un terminal linux, il faut utiliser la commande cd)
- Récupérer l'adresse du dépôt git (« https://gitlab.com/adresse_complète/… »)
- Lancer la commande : **git clone https://gitlab.com/adresse...**
- Easy !

Quelques commandes de base

Autres commandes

- Une fois le dépôt cloné, placez-vous dans le dossier créé après clonage. Toutes les commandes qui vont suivre se feront depuis ce dossier.
- Tirer les sources depuis le serveur délocalisé pour être à jour :
  - **git pull --all**
- Pousser les modifications sur le serveur délocalisé :
  - **git push**
- Créer une Branche :
  - **git checkout -b [name_of_your_new_branch]**
- Se placer sur une branche existante en local :
  - **git checkout [name_of_the_branch]**

Quelques commandes de base

Faire un commit

- Avant de commiter vous pouvez voir les fichiers qui ont été modifiés avec :
  - **git status**
- Il faut ensuite indexer les fichiers modifiés avec :
  - **git add [nom_du_fichier]** // Pour un indexer un seul fichier
  - **git add .** // Pour indexer tous les fichiers modifiés
- Pour commiter les modifications :
  - **git commit -m «message du commit»**
- Il sera ensuite possible de pousser vos modifications avec un git push, mais vous ne pousserez les modifications de votre branche seulement.
- Pour merger votre branche avec la branche Master :
  - **git merge master**

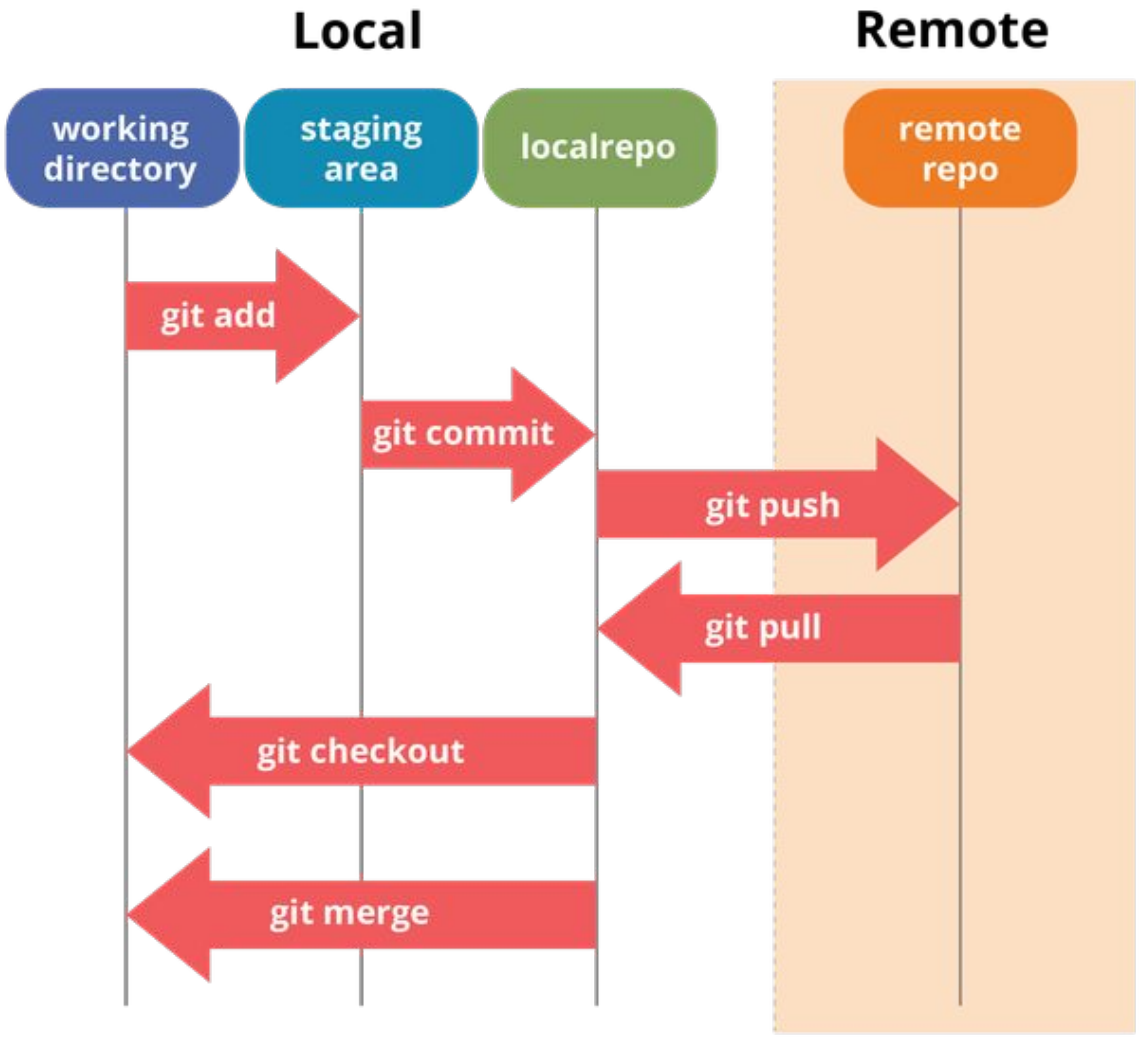
Quelques commandes de base

Bonne pratique :

- Pour pousser vos modifications sur la branche master du serveur délocalisé, il est préconisé de faire les commandes dans l'ordre suivant (en vérifiant en amont que vous êtes sur votre branche et non la branche master) :
  - **Git add .** //Indexer les modifications
  - **Git commit -m «message»** //-> Commit avec message
  - **Git pull --all** //tirer les sources du serveur
  - **Git merge master** //Fusion de notre branche avec master
  - Vérifier qu'il n'y a pas de conflit / Régler les conflits / Refaire les étapes précédentes si des conflits ont été gérés.
  - **Git push** ou **git force push** //Pour pousser notre branche master sur le serveur

- **Attention : au moment du merge, vous allez probablement avoir des conflits qu'il faudra gérer avec un éditeur de texte ou smartgit !!!**

Les différentes zones



INSTITUT NATIONAL
DE L'INFORMATION
GÉOGRAPHIQUE
ET FORESTIÈRE

Merci de votre attention

ign.fr