Git à plusieurs : mode d'emploi

Florian Legendre

Université de Poitiers

Année 2020 - 2021

Table of Contents

Université de Poitiers

Table of Contents

Git n'est pas GitHub !

À l'instar des branches où je considérais que vous aviez un dépôt distant et où je vous montrais comment configurer vos branches locales/distantes depuis Git, ici je vais vous présenter deux dernières notions de Git qui sont indépendantes du choix de dépôt distant que vous ferez.

Ces notions sont celles d'interdictions de push et de conflits d'édition. Bien que le mot "conflit" puisse faire peur il s'agit de quelque-chose de tout à fait ordinaire quand on travaille à plusieurs sur un même projet. Git simplifie grandement la gestion de ces conflits.

Université de Poitiers 1431

Les interdictions de push

Si la branche distante est plus avancée que votre branche locale vous ne pourrez pas mettre vos changements sur la branche distante :

```
crex@crex:~/projects/GitLearn(dev)$ git push
To github.com:Chuxclub/GitLearn.git
 ! [rejected]        dev -> dev (fetch first)
error: failed to push some refs to 'git@github.com:Chuxclub/GitLearn.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

Université de Poitiers 1431

Résolution avec git pull

En lisant bien le message d'erreur précédent vous remarquerez que Git vous suggère tout simplement de d'abord fusionner les changements de la branche distante dans votre branche locale :

```
crex@crex:~/projects/GitLearn(dev)$ git pull
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (4/4), 814 bytes | 58.00 KiB/s, done.
From github.com:Chuxclub/GitLearn
   313203f..5f7b498  dev        -> origin/dev
Merge made by the 'recursive' strategy.
 codes/test.txt | 8 +++++++-
 1 file changed, 7 insertions(+), 1 deletion(-)
```

Résolution avec git pull

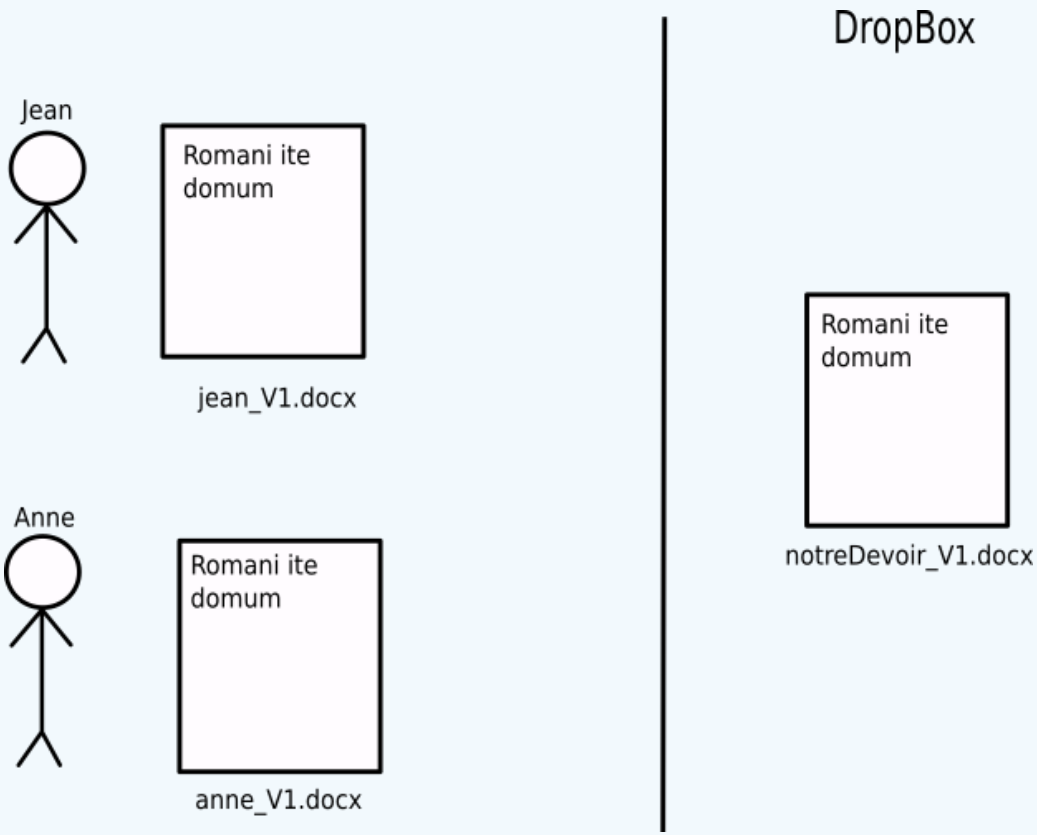
Puis de retenter votre mise à jour de la branche distante :

```
crex@crex:~/projects/GitLearn(dev)$ git push
Enumerating objects: 36, done.
Counting objects: 100% (29/29), done.
Delta compression using up to 8 threads
Compressing objects: 100% (20/20), done.
Writing objects: 100% (20/20), 270.46 KiB | 2.58 MiB/s, done.
Total 20 (delta 14), reused 0 (delta 0)
remote: Resolving deltas: 100% (14/14), completed with 7 local objects.
To github.com:Chuxclub/GitLearn.git
   5f7b498..5684682  dev -> dev
```

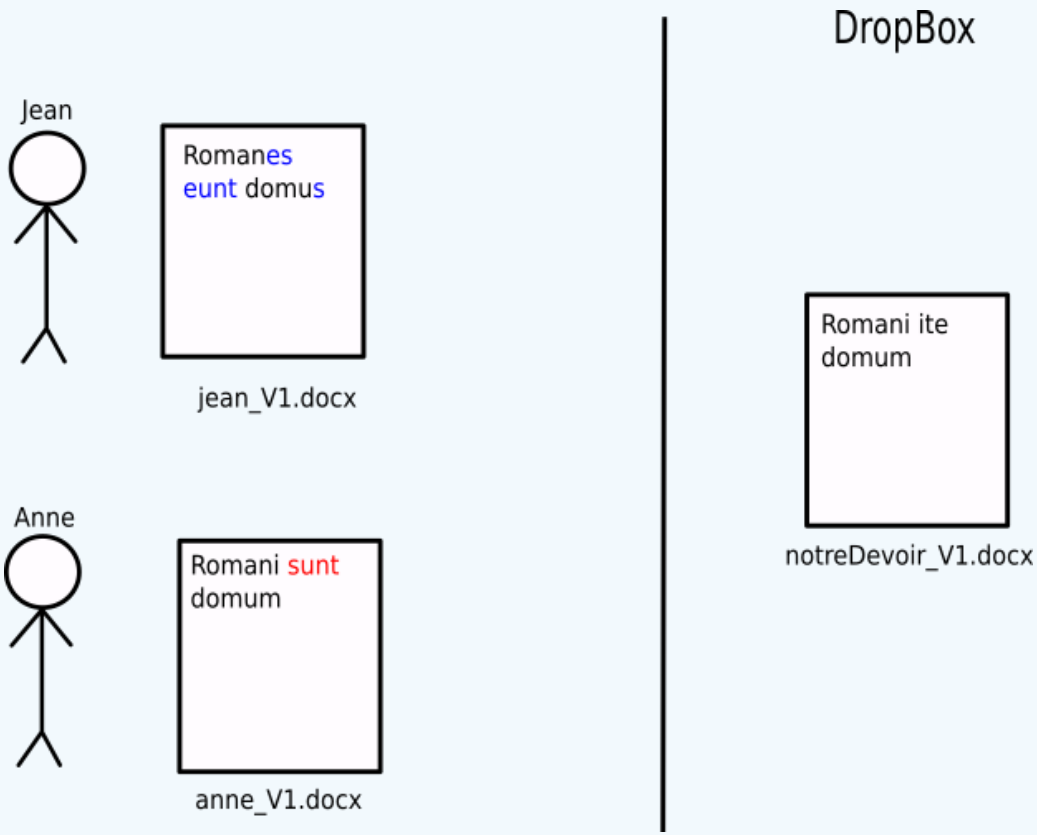
L'idée étant que vous ayiez le même historique de versions que vos collaborateurs !

Table of Contents

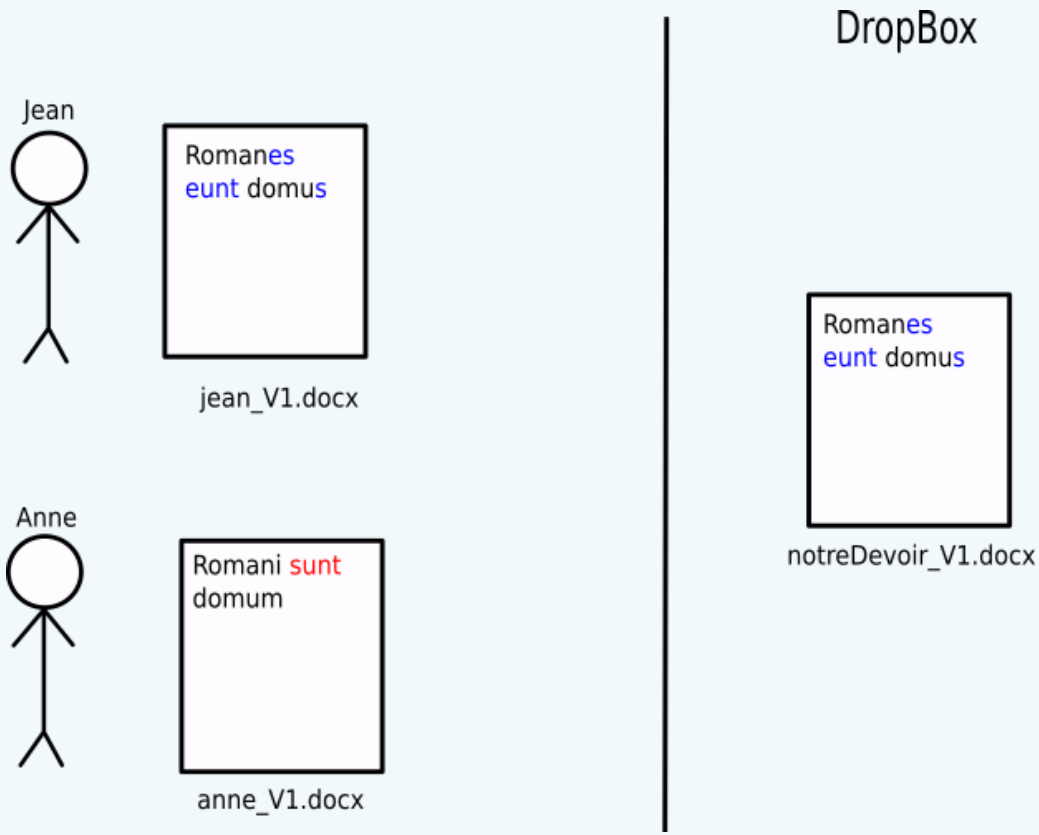
Rappel du problème



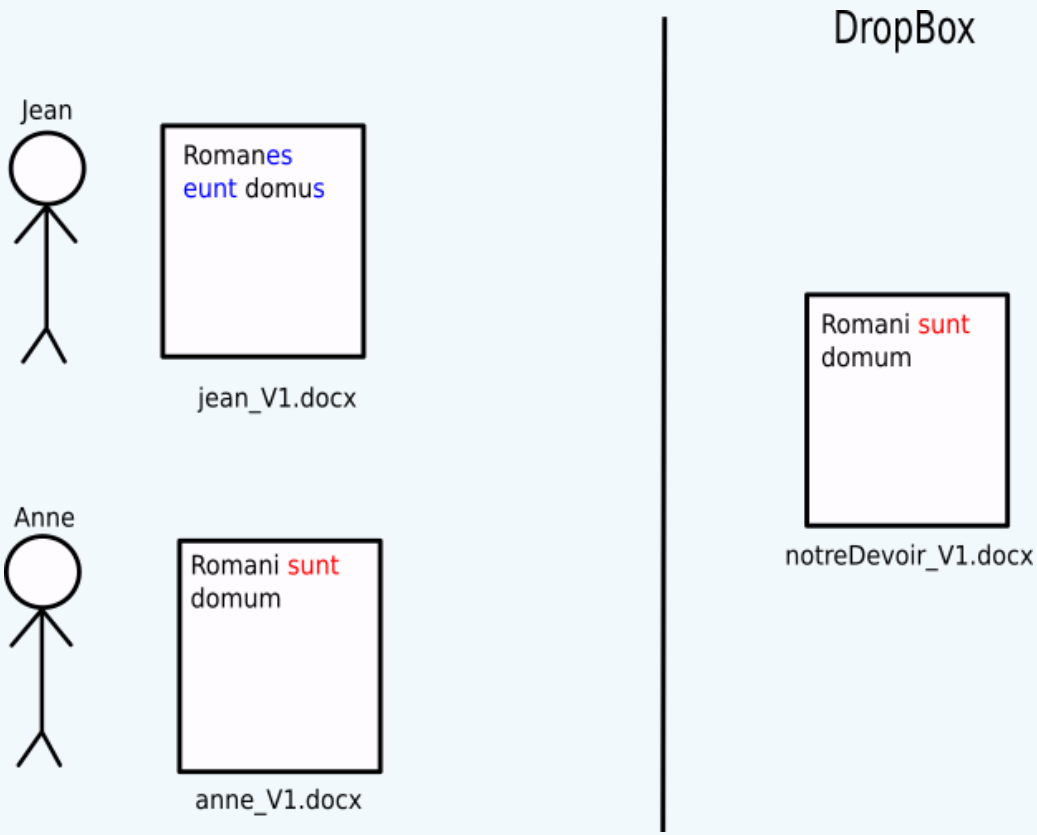
Rappel du problème



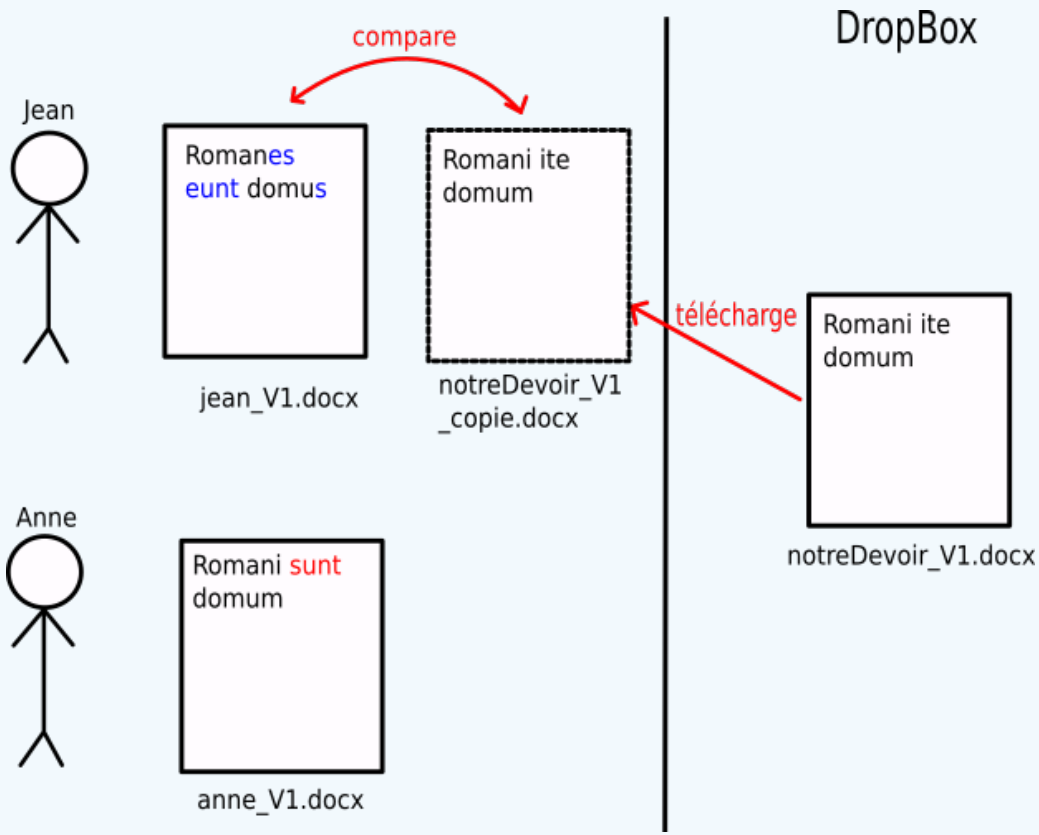
Rappel du problème



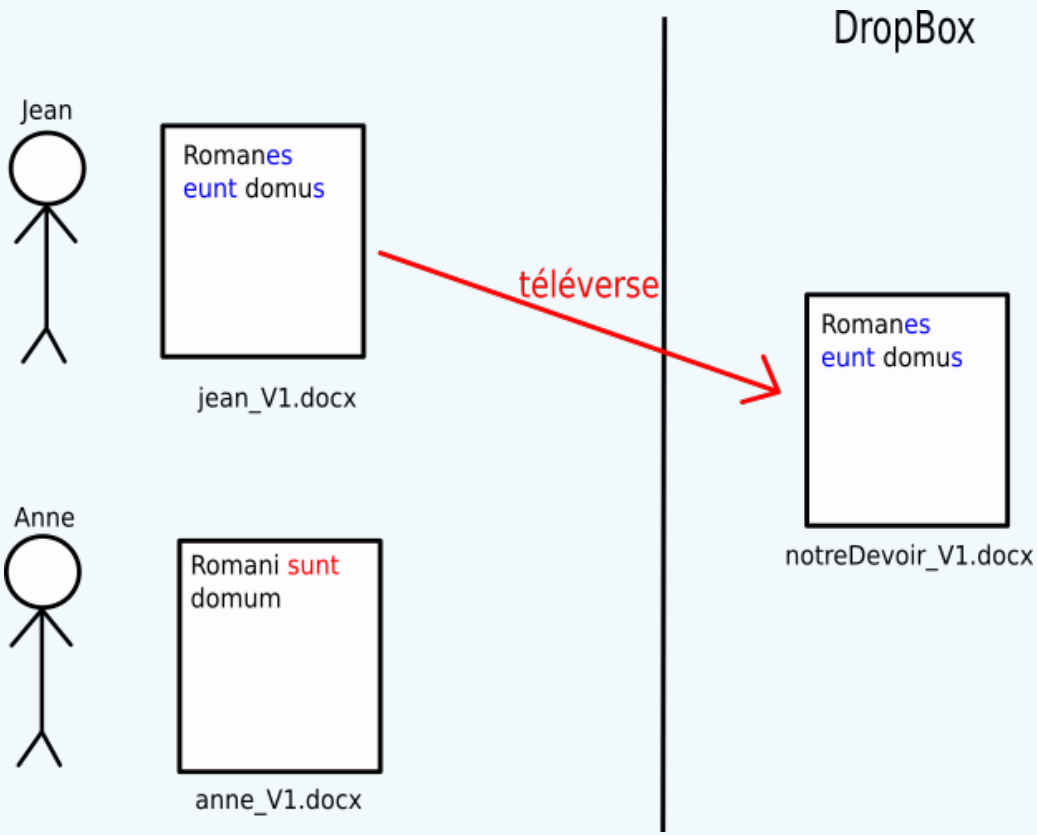
Rappel du problème



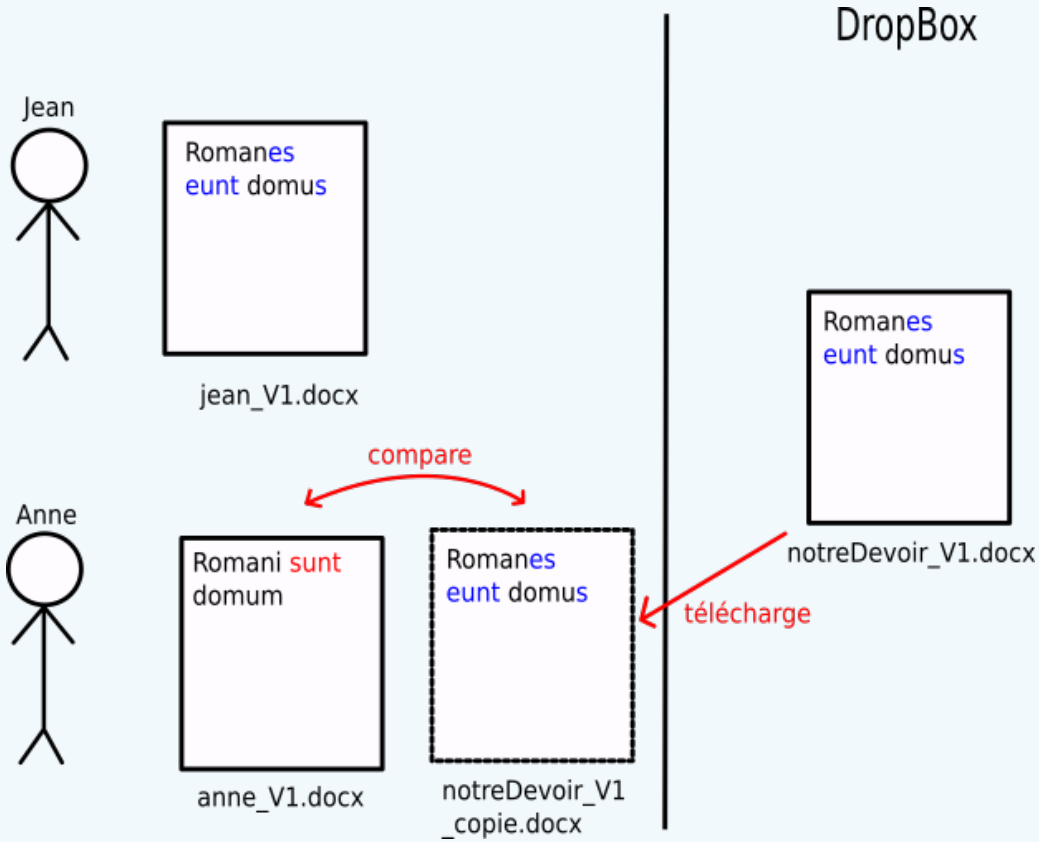
Rappel de la solution "à la main"



Rappel de la solution "à la main"



Rappel de la solution "à la main"



Rôle de Git

Git ne peut pas décider à votre place de ce qui doit être supprimé ou gardé ! Ou alors nous perdons tous notre travail car les ordinateurs peuvent coder...

Mais Git peut vous indiquer dans quels fichiers et à quelles lignes vos deux versions sont en conflit.

Table of Contents

Un conflit détecté par Git

Un conflit survient quand on met à jour sa branche locale avec la branche distante suivie et qu'une ou plusieurs versions de la branche suivie fait état de zones éditées qui sont en conflit avec celles de notre branche locale :

```
crex@crex:~/projects/GitLearn(dev)$ git pull
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (4/4), 705 bytes | 16.00 KiB/s, done.
From github.com:Chuxclub/GitLearn
   a223800..d261500  dev        -> origin/dev
Auto-merging codes/test.txt
CONFLICT (content): Merge conflict in codes/test.txt
Automatic merge failed; fix conflicts and then commit the result.
```

Listing 1 – Exemple de détection automatique de conflit d'édition

Un conflit détecté par Git

Pour savoir quels fichiers sont en conflits vous pouvez aussi utiliser la commande *git status* qui vous renseigne en plus sur la marche à suivre :

```
crex@crex:~/projects/GitLearn(dev)$ git status
On branch dev
Your branch and 'origin/dev' have diverged,
and have 2 and 1 different commits each, respectively.
  (use "git pull" to merge the remote branch into yours)

You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
    both modified:   codes/test.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

Listing 2 – Les fichiers en conflit indiqués par git status

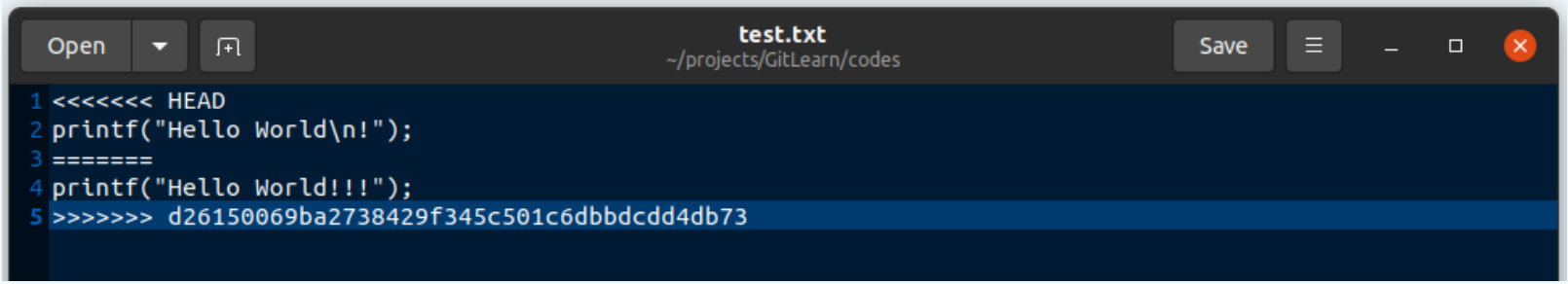
Un conflit détecté par Git

Normalement à chaque fois que vous mettez à jour votre branche locale en faisant un *git pull*, Git crée un commit qui enregistre cette mise à jour (comme ça il est facile de défaire une mise à jour avec un *git revert <id>*).

Cependant, quand Git détecte un conflit il attend que vous ayiez résolu ce-dernier avant de faire un commit de la mise à jour.

Un conflit détecté par Git

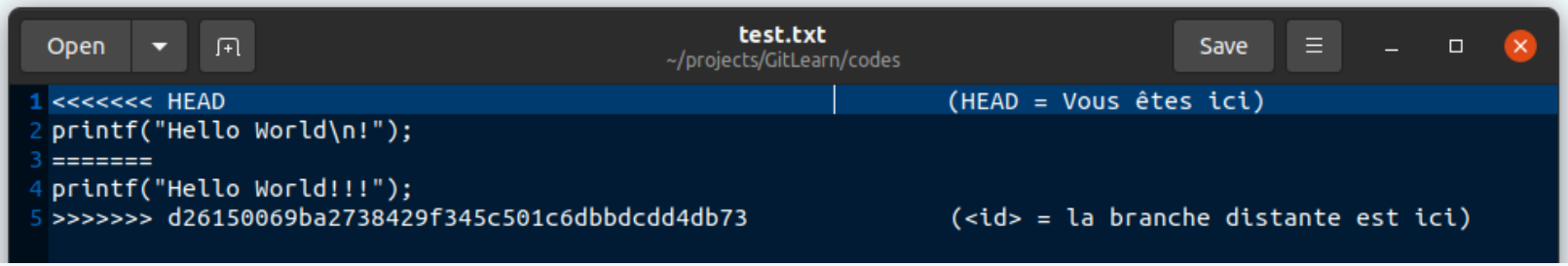
Lors d'un conflit Git **écrit dans** le fichier conflictuel comme ci-dessous :



IMPORTANT : Ne marquez donc pas un fichier comme étant résolu avant d'avoir réédité ce fichier !

Un conflit détecté par Git

Vous remarquerez les deux parties crées par les <<<, === et >>> de Git. La première partie (HEAD) correspond à ce qu'il y a chez vous (Rappel : HEAD = "vous êtes ici pour Git"), la seconde partie correspond à ce qu'il y a sur la branche distante :

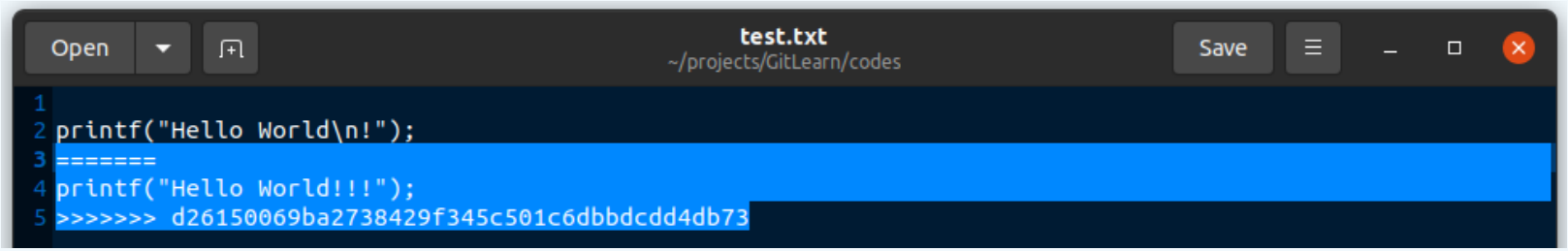


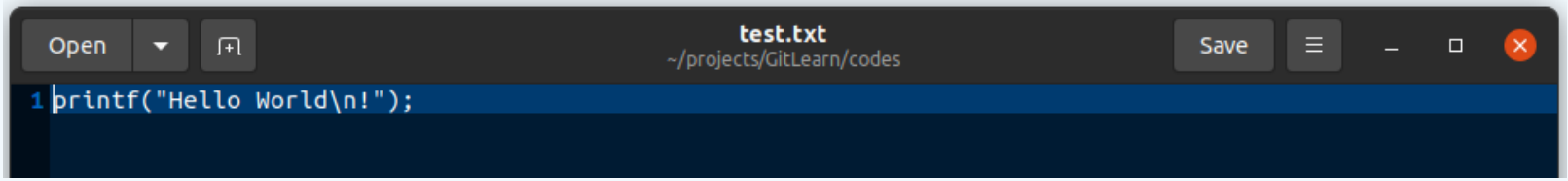
Vous faites la résolution

Git vous a épargné la détection des conflits. Mais la gestion/résolution reste à votre charge ! Pour ce faire il suffit d'aller dans le fichier conflictuelle/édité par Git et de supprimer/garder les lignes qui vous intéressent.

Vous faites la résolution

Je peux faire ce que je veux lors d'une édition de conflits : garder tout ou partie ou rien du tout du travail de mon collègue ou le mien. Vous faites ce choix selon vos besoins. Ici je choisis de supprimer entièrement le travail qui était sur la branche distante :





Vous faites la résolution

Une fois que vous avez fini de choisir ce que vous vouliez garder ou non (discutez-en avec la personne concernée avant !) vous pouvez ajouter les fichiers au staging avec *git add <nom_fichier>* afin de préparer le commit que Git a suspendu en vous attendant :

```
crex@crex:~/projects/GitLearn(dev)$ git add -A
crex@crex:~/projects/GitLearn(dev)$ git status
On branch dev
Your branch and 'origin/dev' have diverged,
and have 2 and 1 different commits each, respectively.
  (use "git pull" to merge the remote branch into yours)

All conflicts fixed but you are still merging.
  (use "git commit" to conclude merge)

Changes to be committed:
    modified:   codes/test.txt
crex@crex:~/projects/GitLearn(dev)$ git commit -am "Conflit résolu"
[dev 313203f] Conflit résolu
```

Listing 3 – Fin de résolution du conflit