GIT ET LA GESTION DE VERSIONS 3/3

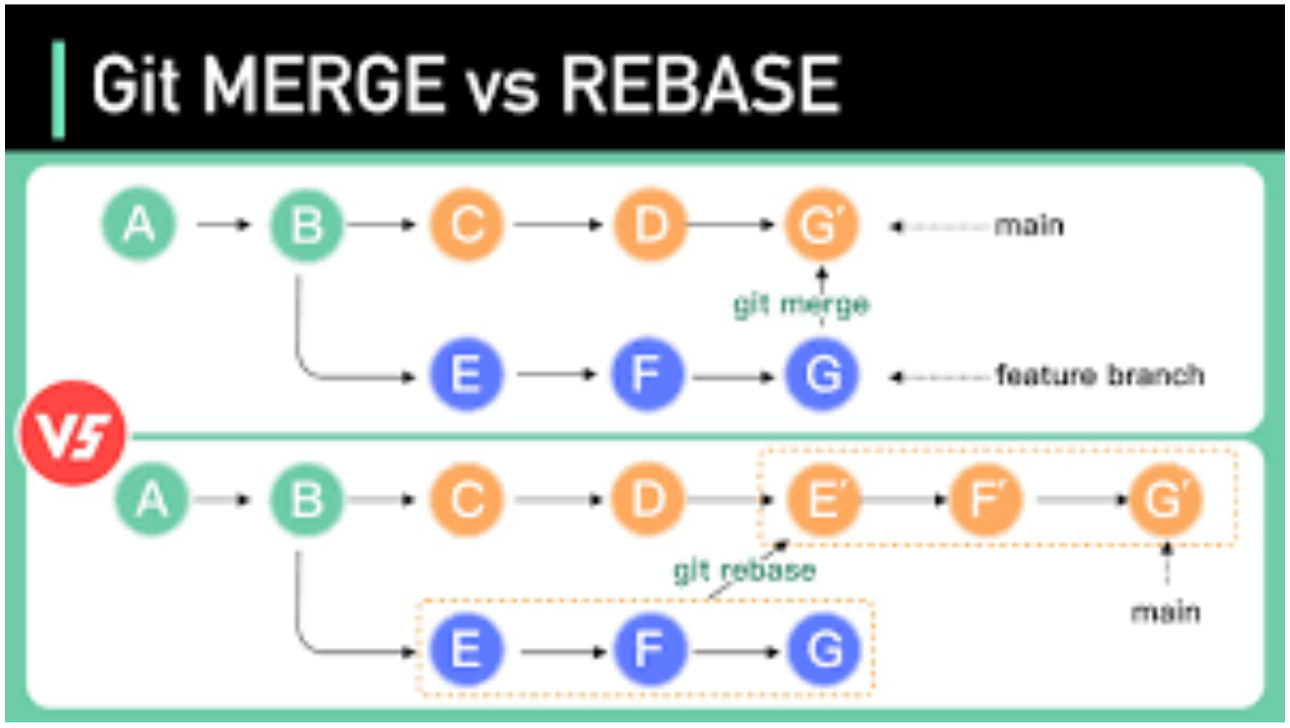
Auteur : Sébastien Inion

Gestion de l’historique de Git

Différences fondamentales :

- *Merge* : Combine deux branches en créant un commit de fusion (*merge commit*).

- *Rebase* : Déplace les commits d’une branche pour qu’ils soient intégrés dans une autre sans créer de commit de fusion.



MERGE

Merge

Le merge est une opération Git utilisée pour combiner deux branches distinctes en une seule. Cette action intègre les modifications apportées dans une branche (souvent une branche de fonctionnalité ou une branche secondaire) dans une autre branche (généralement la branche principale comme main ou master).

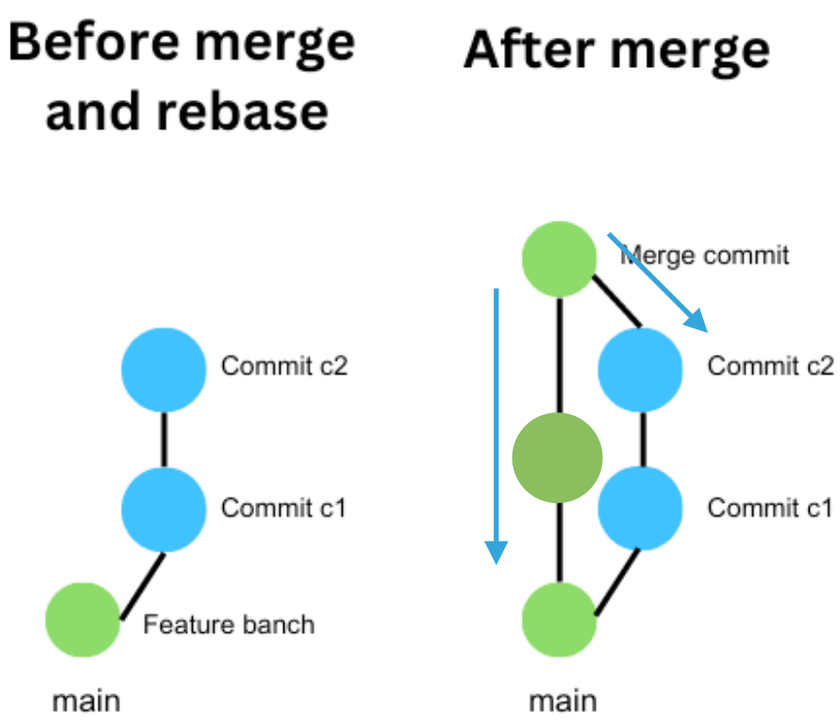
Branches impliquées :

Branche cible : Celle dans laquelle vous voulez intégrer les changements (par exemple, main).
Branche source : Celle dont les changements seront fusionnés (par exemple, feature-branch).

Merge

Création d'un commit de merge :

- Lors d'un merge, Git crée un commit de merge si les branches ont divergé.
- Ce commit inclut les modifications des deux branches et pointe vers les deux branches parentes.
- Il préserve tout l'historique des deux branches.



Merge

- Il y a deux cas possibles avec le git merge

Fast-Forward Merge :

- Si aucun nouveau commit n'a été ajouté à `main` après `Commit c1`
- Git avance simplement le pointeur de `main` pour inclure tous les commits de `feature-branch`.

main: c1
feature: \---c2---c3

Après le merge :
main: c1---c2---c3

Merge

On ajoute index.html sur la branche main et on commit

```
(base) MacBook-2:testsGitStudi sebastien$ touch index.html
(base) MacBook-2:testsGitStudi sebastien$ git add index.html
(base) MacBook-2:testsGitStudi sebastien$ git commit -m "Ajout du fichier index.html"
[main e5a5465] Ajout du fichier index.html
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 index.html
(base) MacBook-2:testsGitStudi sebastien$ git log -n 1
commit e5a54651a4174c31be4e3cece54cb6a68e76b176 (HEAD -> main)
Author: SebInfo <contact@inion.info>
Date:   Mon Jan 20 09:41:44 2025 +0100

    Ajout du fichier index.html
```

Merge

On va créer une branche feature ajouter deux fichiers et faire deux commit

```
(base) MacBook-2:testsGitStudi sebastien$ git branch feature
(base) MacBook-2:testsGitStudi sebastien$ git checkout feature
Basculement sur la branche 'feature'
(base) MacBook-2:testsGitStudi sebastien$ touch inscription.php
(base) MacBook-2:testsGitStudi sebastien$ git add inscription.php
(base) MacBook-2:testsGitStudi sebastien$ git commit -m "Ajout du formulaire d'inscription"
[feature 1cd963d] Ajout du formulaire d'inscription
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 inscription.php
(base) MacBook-2:testsGitStudi sebastien$ touch contact.php
(base) MacBook-2:testsGitStudi sebastien$ git add contact.php
(base) MacBook-2:testsGitStudi sebastien$ git commit -m "Ajout de la page contact"
[feature c35449d] Ajout de la page contact
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 contact.php
```

Merge

On merge depuis la branche principale

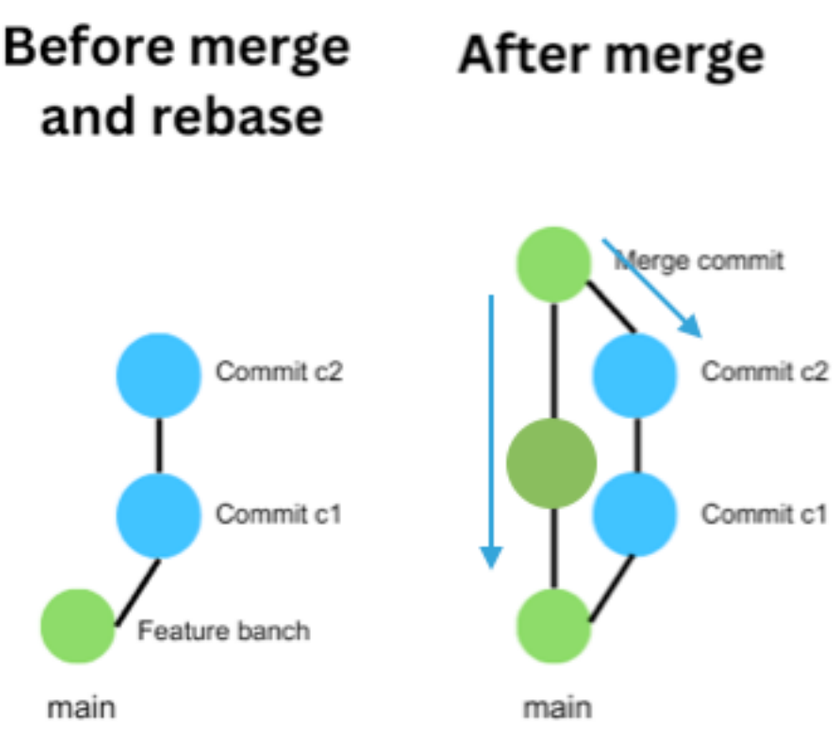
```
(base) MacBook-2:testsGitStudi sebastien$ git checkout main
Basculement sur la branche 'main'
Votre branche est en avance sur 'monDepot/main' de 1 commit.
  (utilisez "git push" pour publier vos commits locaux)
(base) MacBook-2:testsGitStudi sebastien$ git merge feature
Mise à jour e5a5465..c35449d
Fast-forward
 contact.php     | 0
 inscription.php | 0
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 contact.php
 create mode 100644 inscription.php
(base) MacBook-2:testsGitStudi sebastien$ git log -n 4
commit c35449d752423079dca132dc33c0820e9f871611 (HEAD -> main, feature)
Author: SebInfo <contact@inion.info>
Date:   Mon Jan 20 09:47:24 2025 +0100

    Ajout de la page contact

commit 1cd963d3610f1548fe223b5d99f3eec70bf8178f
Author: SebInfo <contact@inion.info>
Date:   Mon Jan 20 09:46:34 2025 +0100

    Ajout du formulaire d'inscription

commit e5a54651a4174c31be4e3cece54cb6a68e76b176
Author: SebInfo <contact@inion.info>
Date:   Mon Jan 20 09:41:44 2025 +0100

    Ajout du fichier index.html
:...skipping...
commit c35449d752423079dca132dc33c0820e9f871611 (HEAD -> main, feature)
```

Merge

- Il y a deux cas possibles avec le git merge

Merge avec un commit de fusion :

- Si des commit on été ajouté sur le main depuis la création de la branche
- Git combine les deux branches en créant un **commit de merge**.



Merge

```
(base) MacBook-2:testsGitStudi sebastien$ git commit -m "Ajout du panier"
[main 743086e] Ajout du panier
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 panier.php
(base) MacBook-2:testsGitStudi sebastien$ git log -n 1
commit 743086e059d8671bbc2c8cee45a025d375b6eba2 (HEAD -> main)
Author: SebInfo <contact@inion.info>
Date:   Mon Jan 20 10:03:06 2025 +0100

    Ajout du panier
```

Merge

```
(base) MacBook-2:testsGitStudi sebastien$ git branch feature2
(base) MacBook-2:testsGitStudi sebastien$ git checkout feature2
Basculement sur la branche 'feature2'
(base) MacBook-2:testsGitStudi sebastien$ touch secure.php
(base) MacBook-2:testsGitStudi sebastien$ git add secure.php
(base) MacBook-2:testsGitStudi sebastien$ git commit -m "Ajout du fichier secure"
[feature2 773ea10] Ajout du fichier secure
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 secure.php
```

Merge

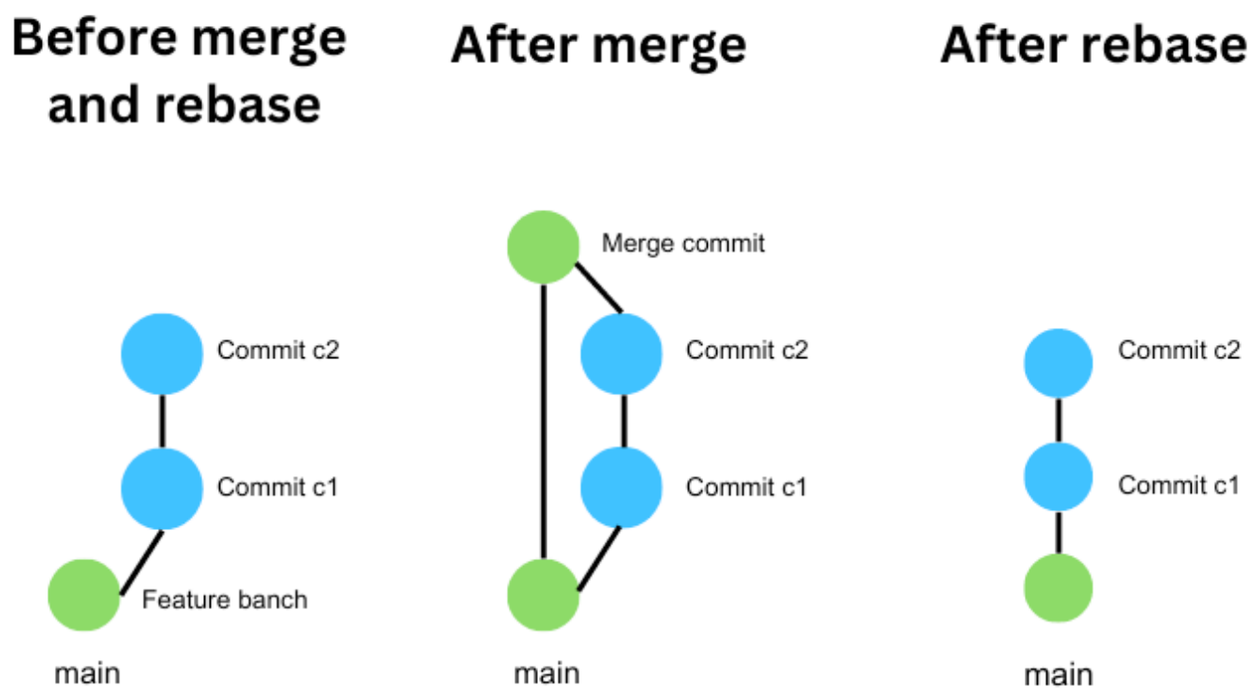
```
(base) MacBook-2:testsGitStudi sebastien$ git checkout main
Basculement sur la branche 'main'
Votre branche est en avance sur 'monDepot/main' de 4 commits.
  (utilisez "git push" pour publier vos commits locaux)
(base) MacBook-2:testsGitStudi sebastien$ touch recherche.php
(base) MacBook-2:testsGitStudi sebastien$ git add recherche.php
(base) MacBook-2:testsGitStudi sebastien$ git commit -m "Ajout recherche"
[main a2f8126] Ajout recherche
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 recherche.php
(base) MacBook-2:testsGitStudi sebastien$ git merge feature2
Merge made by the 'ort' strategy.
 secure.php | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 secure.php
(base) MacBook-2:testsGitStudi sebastien$ git log -n 4
commit 2fb47573b499427f18310a68eaf6024ec497cb27 (HEAD -> main)
Merge: a2f8126 773ea10
Author: SebInfo <contact@inion.info>
Date:   Mon Jan 20 10:06:56 2025 +0100

    Fusion de la branche feature2 dans main
    Ajout de la fonctionnalité X terminéeMerge branch 'feature2'

commit a2f812660707d6183a1a97bfb5746a053cd11a85
Author: SebInfo <contact@inion.info>
```

REBASE

Rebase

Le **rebase** (ou "rebasage" en français) est une commande Git permettant de **réappliquer des commits** d'une branche sur une autre, de manière linéaire, en modifiant l'historique des commits.

Contrairement à un **merge**, qui crée un commit de fusion pour intégrer deux branches, un **rebase** réécrit l'historique de la branche pour la rendre plus propre et linéaire.



Rebase

Quand utiliser le rebase ?

- **Maintenir un historique linéaire** : Si tu veux éviter les commits de fusion, le rebase peut être utilisé pour "rejouer" les commits d'une branche sur une autre.
- **Mettre à jour une branche avec les dernières modifications** : Lorsqu'une branche est dérivée d'une autre, un rebase permet de réappliquer les changements sur la base la plus récente.

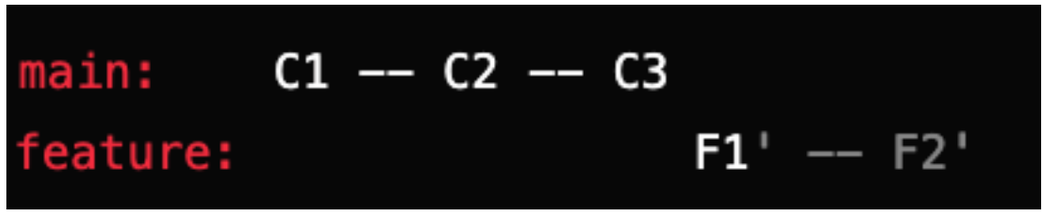
On ne perd pas l’autre branche mais l’historique est réécrit !

Rebase

Supposons deux branches, `main` et `feature` :

- `main` contient les commits : `C1`, `C2`, `C3`.
- `feature` contient les commits : `F1`, `F2`. (F1 branche après C1)

Après :
git checkout feature
git rebase main

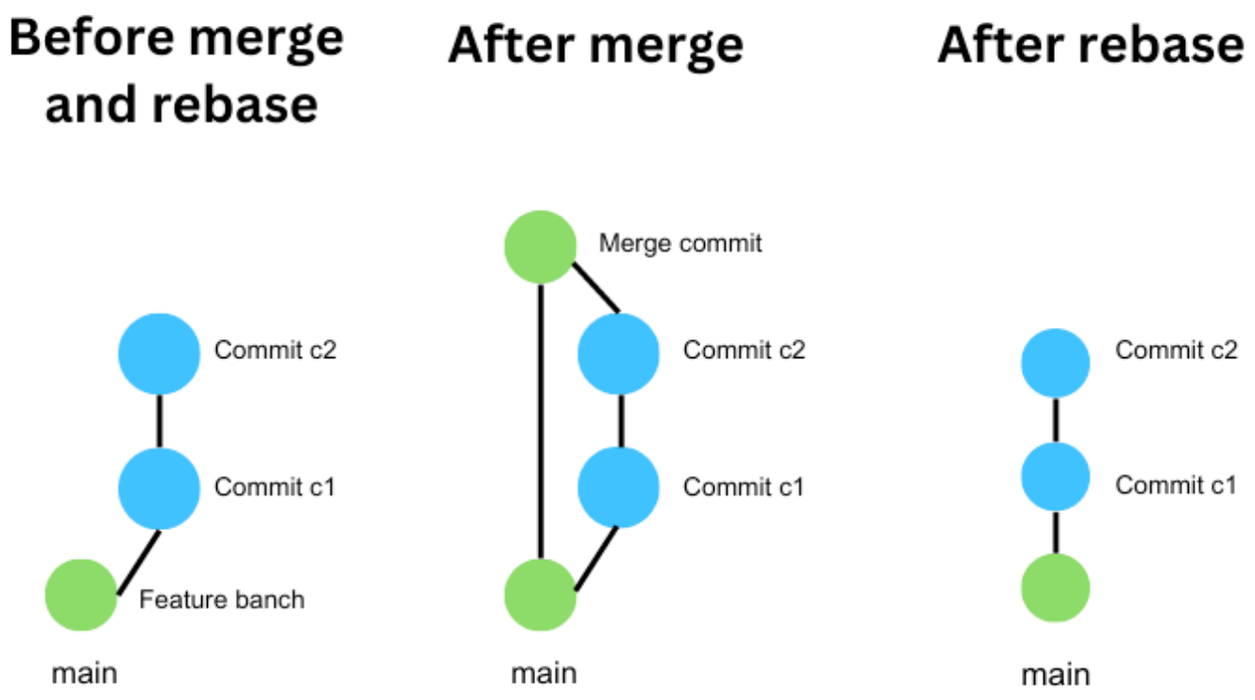


Les commits `F1'` et `F2'` sont des **nouvelles versions** des commits `F1` et `F2` mais basées sur `C3`.

Rebase

- **Sur une branche partagée avec d'autres développeurs** : Comme le rebase réécrit l'historique, cela peut causer des problèmes si d'autres utilisateurs ont déjà basé leur travail sur l'ancienne version de la branche.

Le rebase est un outil puissant, mais il doit être utilisé avec précaution !



MERGE/REBASE

Merge vs Rebase

Merge :

Avantages :

Historique préservé : Le merge conserve l'historique complet des deux branches, y compris les commits divergents.

Simple et sûr : Pas de risque de réécrire l'historique existant, ce qui est idéal pour des projets collaboratifs ou publics.

Rapide : Fusionne les branches sans modifier les commits existants.

Inconvénients :

Historique plus complexe : Introduit des "branches" et des commits de merge qui peuvent encombrer l'historique, surtout si les branches sont courtes ou nombreuses.

Merge vs Rebase

Rebase :

Avantages :

Historique propre et linéaire : Idéal pour une meilleure lisibilité, notamment dans les projets où l'historique est important pour les analyses.

Intégration fluide : Les commits d'une branche sont rejoués sur la branche cible, comme s'ils avaient été créés directement sur celle-ci.

Inconvénients :

Risque de réécriture de l'historique : Si utilisé sur des branches partagées ou publiques, cela peut entraîner des conflits pour d'autres développeurs.

Plus complexe : Nécessite une bonne compréhension de Git pour gérer les éventuels conflits lors du rebase.

Merge vs Rebase

Quand utiliser Merge :

Lors de la collaboration avec plusieurs développeurs.
Pour intégrer des branches sans modifier leur historique.
Lorsqu'il est important de conserver la trace des commits individuels et des fusions.

Quand utiliser Rebase :

Pour un développement individuel ou des branches locales.
Lorsqu’un historique linéaire et propre est souhaité.
Avant de soumettre des modifications à une branche principale (après un pull ou avant un merge).

MERCI POUR VOTRE ATTENTION