

Formation Git

VI - Les branches

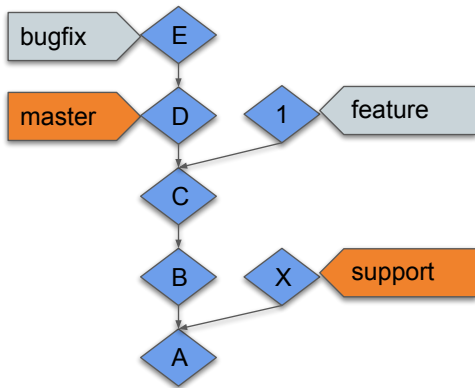


Arnaud MERCIER
arnaud.mercier@hexotech.fr



Nous allons maintenant voir ce qui a fait de Git l'outil de gestion de conf le plus populaire de nos jours: les branches
Plus précisément la simplicité et la flexibilité de son système de branche

Le système de branches



Branches temporaires

- branches de travail
- développement de fonctionnalités
- corrections de bug

Branches permanentes

- contient les release
- branche principale
- branche de support de version

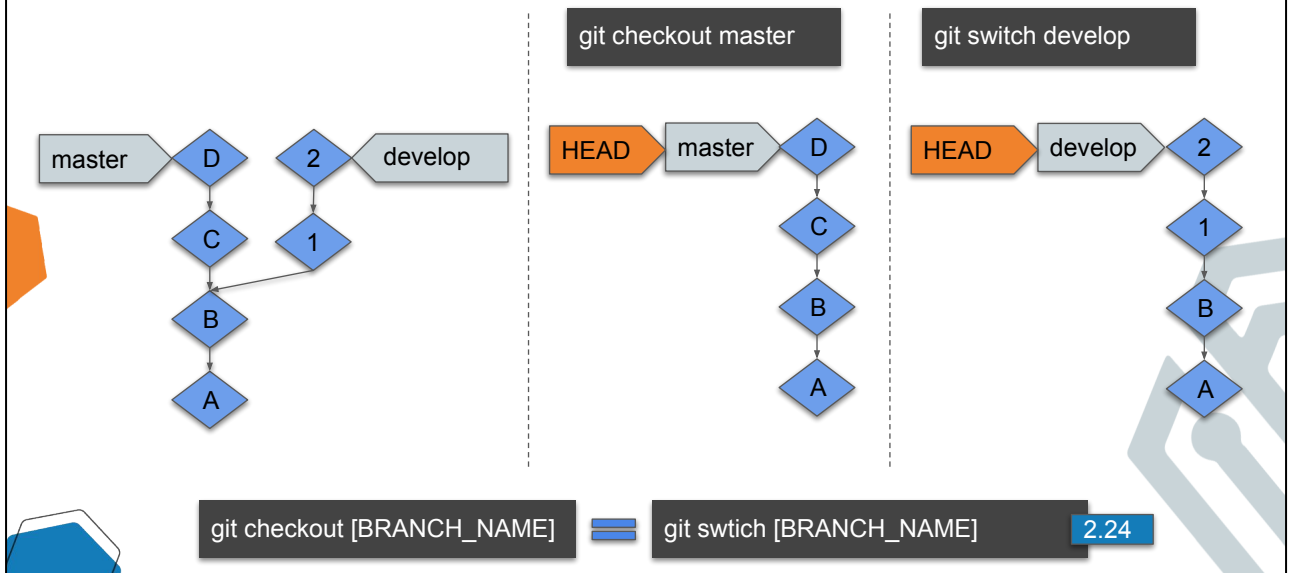
les branches permettent d'avoir plusieurs historiques en parallèles. Il existe 2 catégories de branche:

- Branches permanentes; utilisés porter les commits de release soit via la branche principale soit via des branches de support de version
- Branches temporaires; utilisés pour réaliser un développement ou une correction de bug. Cette branche est supprimé une fois mergée

Exemple (DEPOT):

- **git init**
- **touch source.txt**
- **Ajouter une lettre par ligne et faire un commit a chaque fois pour avoir l'historique de master du slide**

Le système de branches

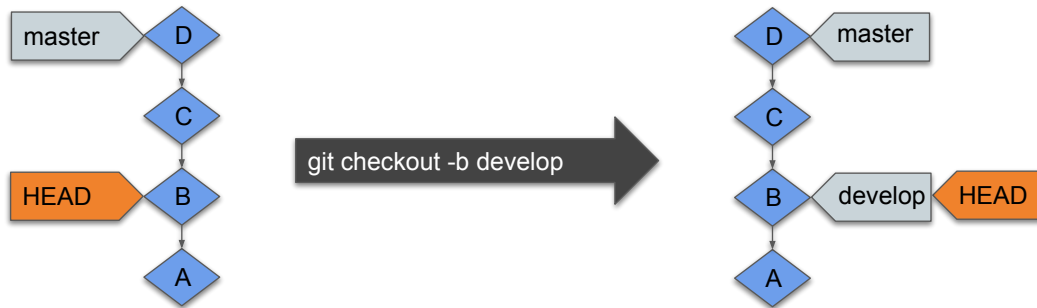


Une branche est simplement un pointeur, sur un commit. Une branche est alors constitué de ce commit pointé et de ses parents.

Exemple (DEPOT):

- `gitk -all`
- `git switch master`
- `git log --oneline`

Créer une branche



```
git branch [branch_name] [sha1/ref]
git checkout [branch_name]
```

```
git checkout -b [branch_name] [sha1/ref]
```

```
git branch [branch_name] [sha1/ref]
git switch [branch_name]
```

```
git switch -c [branch_name] [sha1/ref] git 2.24.0
```

Pour créer une branche, il faut utiliser la commande `git branch "[branch_name] [sha1/ref]"`

Il est aussi possible de se déplacer sur une branche et la créer si elle n'existe pas encore via la commande: `git switch -c [branch_name] [sha1/ref]`

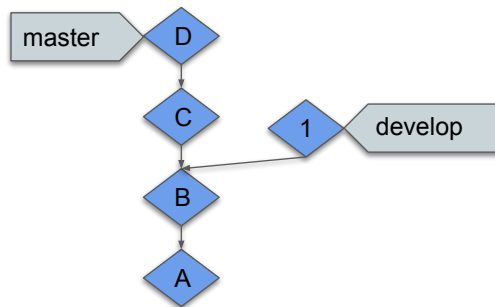
Il peut y avoir plusieurs branches sur un même commit

INFO: Master n'est pas une branche différente, elle est la branche principale par convention seulement

Exemple (DEPOT):

- `git branch TOTO`
- `git switch TOTO`
- ajout de la ligne E dans le fichier source.txt
- `git commit -am"E"`
- `gitk -all`
- `git switch -c master~2`
- ajout de la ligne 1 dans le fichier source.txt
- `git commit -am"1"`
- ajout de la ligne 2 dans le fichier source.txt
- `git commit -am"2"`
- `gitk -all`

Lister les branches locales



```
git branch
```

```
* master  
develop
```

```
git branch
```

```
* (HEAD detached at 212fa4f)  
develop
```

```
git branch [-v] // Affiche les branches locales
```

Pour lister les branches du dépôt local, il faut utiliser la commande “git branch”
l’option -v permet d’afficher plus d’informations sur la branche et son commit pointé

* devant le nom de branche, permet de voir quelle est la branche courante

INFO: la branche courante est également affichée dans “git status”

Voir l'historique de plusieurs branches

```
commit 8a16f0301c192603761047146df513de21b985c2 (HEAD -> master, tag: MON_SITE_V3)
Author: Arnaud Mercier <arnaud.mercier.formatio@gmail.com>
Date: Wed Jun 13 19:05:35 2018 +0200

    ajout médaille d'or

commit bd88f16cf89d6c2b2bd6f0776567fb915683a70 (origin/master)
Merge: 92df0e3 6d73307
Author: clone <arnaud.mercier.formatio@gmail.com>
Date: Wed Jun 13 17:47:54 2018 +0200

    merge de ma branche BRANCH A MERGE avec ma master

commit 6d73307cb5c9581d802816ee455c73d631eb3385 (origin/BRANCH_A_MERGE)
Author: clone <arnaud.mercier.formatio@gmail.com>
Date: Wed Jun 13 17:20:01 2018 +0200

    ajout git log

commit 5f0a57681501bc3bce4707c5deaf45ea07a994 (BRANCH_A_MERGE)
Author: Arnaud Mercier <arnaud.mercier.formatio@gmail.com>
Date: Wed Jun 13 17:06:28 2018 +0200

    ajout git push

commit d4d4451ec7cf1186e312c341d91a0a3d0fa44665
Author: Arnaud Mercier <arnaud.mercier.formatio@gmail.com>
Date: Wed Jun 13 17:04:28 2018 +0200

    ajout commande git pull

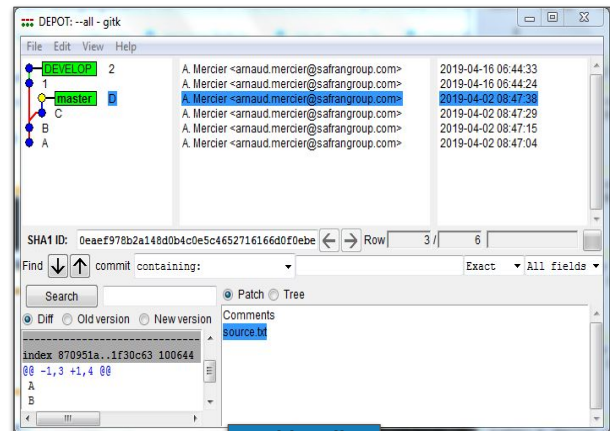
commit 92df0e310f1bce1023e8aecdf7e78d7a919dbba9
Author: Arnaud Mercier <arnaud.mercier.formatio@gmail.com>
Date: Wed Jun 13 17:04:28 2018 +0200

    ajout commande git pull

commit e0a68a7debb02994696bd0d04719b3e797cbe14 (tag: MON_SITE_V4)
Author: clone <arnaud.mercier.formatio@gmail.com>
Date: Wed Jun 13 16:16:56 2018 +0200

    ajout médaille argent
```

git log --graph --all

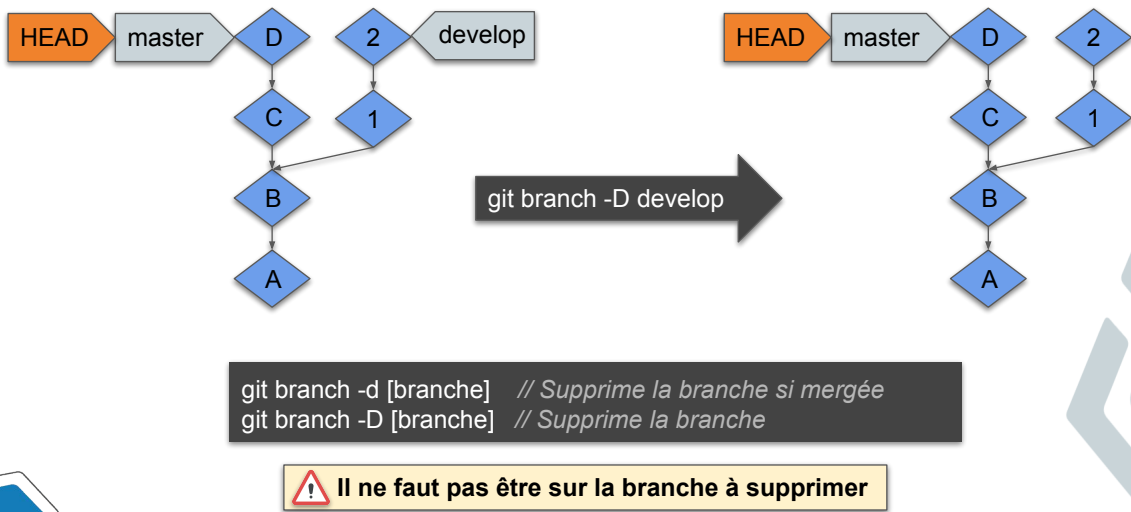


gitk --all

```
git log [sha1 / ref] // Par défaut HEAD
git log --graph --all // Voir sous forme de graph les branches
git log --oneline --decorate --graph --all
```

Pour afficher l'historique d'un dépôt, il faut utiliser la commande "git log". Il est possible d'ajouter l'option "--graph" pour un affichage plus graphique.

Supprimer une branche



Il est possible de supprimer une branche en local via la commande “git branch -d/-D” suivi du nom de la branche.

l’option -d vérifie que la branche est merge. Si ce n’est pas le cas, la suppression n’est pas appliqué

l’option -D ne vérifie pas que la branche est merge

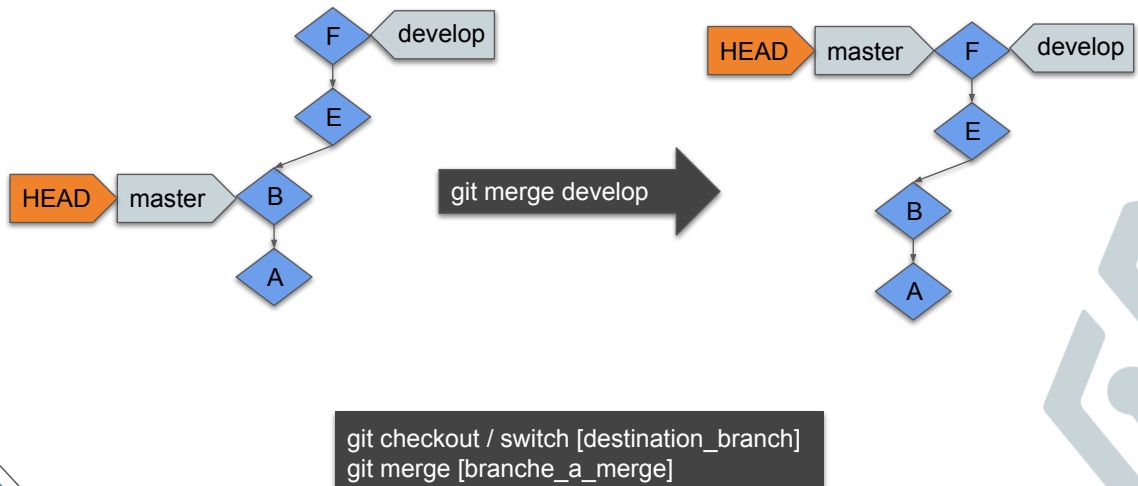
Attention: supprimer une branche correspond au fait de supprimer la référence et non pas les commits. Lors de la suppression de la branche git affiche le sha1 du commit qui était pointé. Il est alors possible de recréer une branche sur le commit en question

Info: Il n’est pas possible de supprimer la branche active. Il faut la désactiver avant

Exemple (DEPOT):

- **git branch -d TOTO**
- **git branch TOTO [sha1]**

Faire un merge de branches: **fast-forward**



Si l'historique entre les deux branches à merge est linéaire (l'une à la suite de l'autre), alors un simple fast-forward est appliqué. Cela signifie que la branche courante va remonter les commits pour venir au même niveau que la branche à merge.

Le merge en fast forward n'entraîne pas de commit de merge ou de conflits.

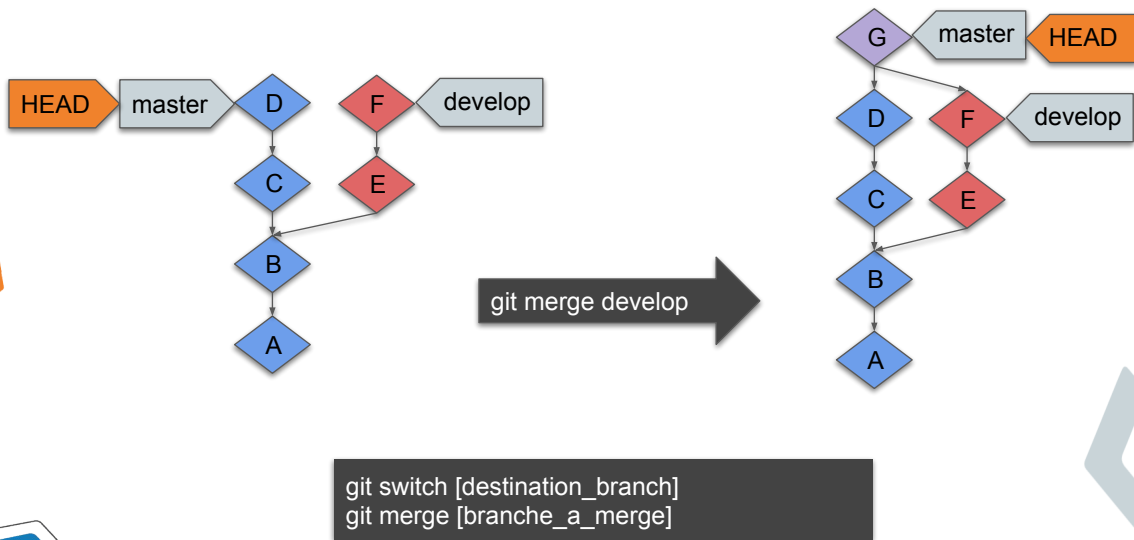
Attention: pour ne pas vous tromper de sens dans le merge, il faut toujours activer la branche à modifier, puis merge l'autre. Par exemple, pour merge la branche de travail develop dans la branche principale master, il faut:

- switch master
- merge develop

Exemple (DEPOT):

- gitk --all
- git switch master
- git merge TOTO
- gitk --all

Faire un merge de branches: **commit de merge**



Lorsque les 2 branches d'un merge ne sont pas en fast forward, un commit de merge est alors obligatoire. Cela permet d'avoir les deux historiques concurrent qui se retrouve via un commit dans la branche courante.

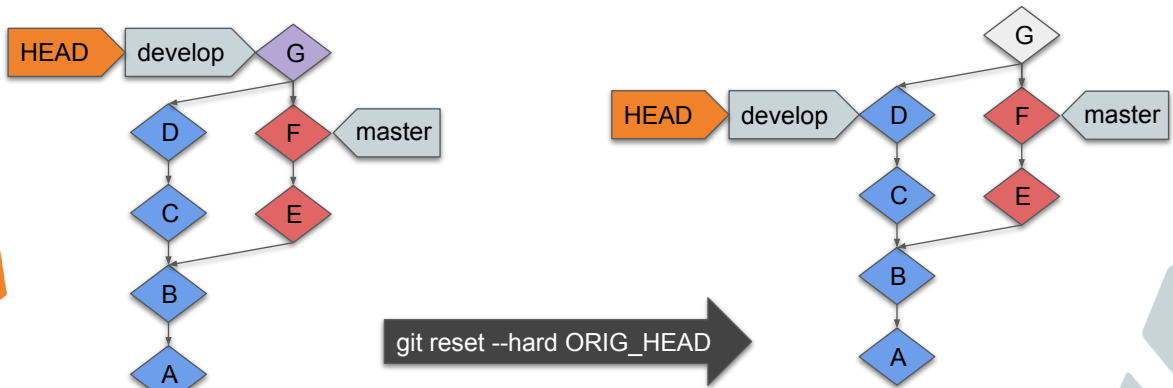
Attention: pour ne pas vous tromper de sens dans le merge, il faut toujours activer la branche a modifier, puis merge l'autre. Par exemple, pour merge la branche de travail develop dans la branche principale master, il faut:

- switch master
- merge develop

Exemple (DEPOT):

- gitk --all
- git switch master
- git merge develop
- gitk --all

Annuler un merge



⚠ uniquement si aucune grosse action depuis le merge

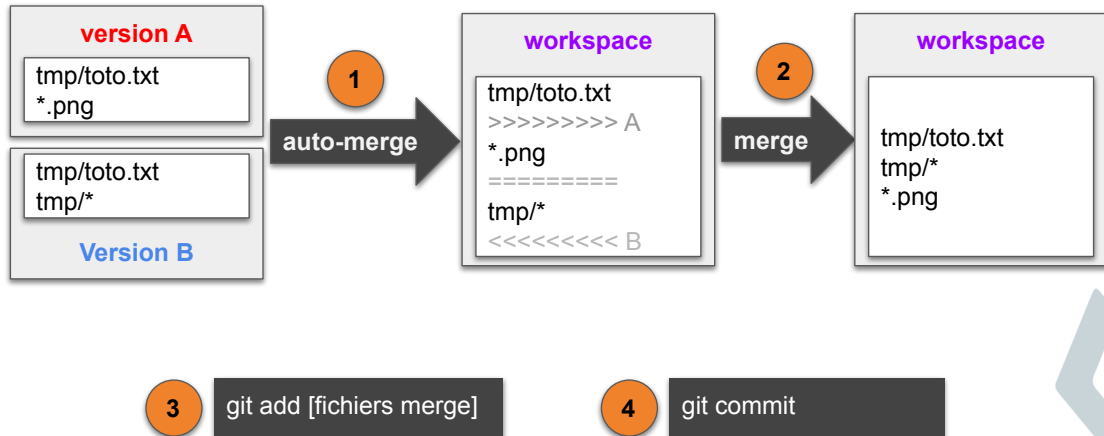
Oups, j'ai fait mon merge à l'envers: exemple, j'ai merge master sur develop au lieu de l'inverse et pas de chance ça a fonctionné.

il est possible d'annuler l'action du rebase avec la commande "git reset --hard ORIG_HEAD" plus d'info au chapitre sur git Reset

Exemple (DEPOT):

- `git reset --hard ORIG_HEAD`
- `gitk --all`

Résoudre un conflit de merge



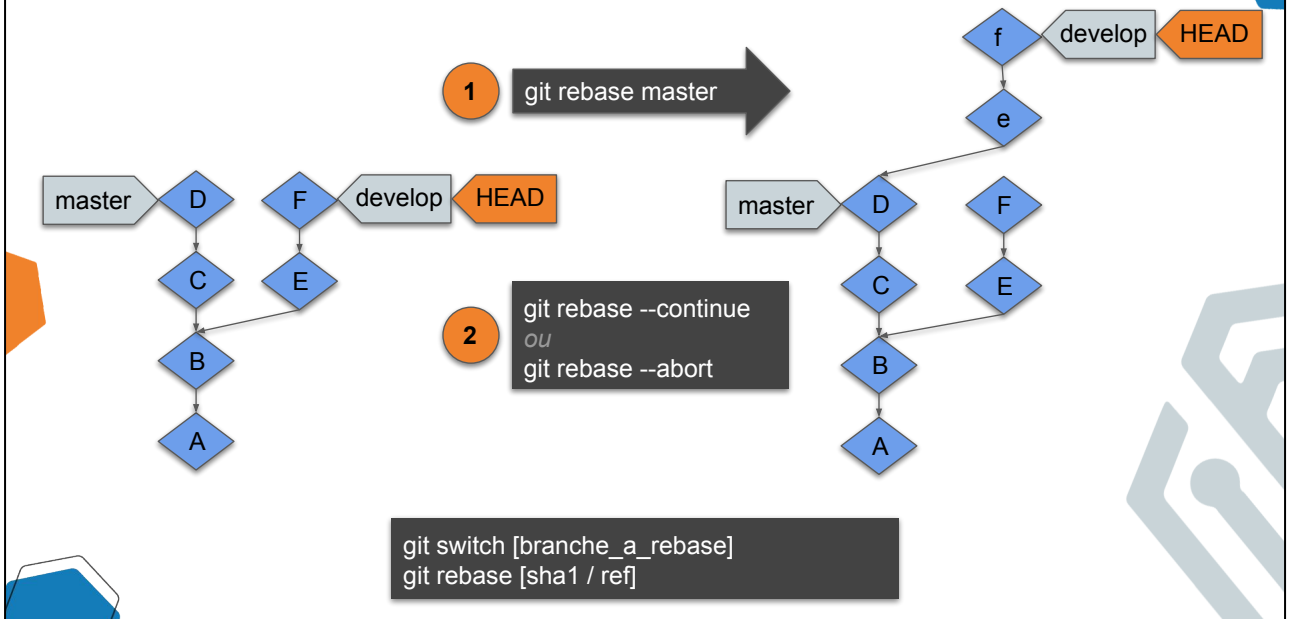
Un commit de merge peut entraîner des conflits. Cela se produit si une même portion de code est modifiée dans les 2 branches. Dans ce cas, git ne sais pas résoudre lui même le merge. Il vous faut alors faire le merge de la partie concerné puis d'indexer les modifications afin de marque comme résolu le conflit. Enfin, il ne reste plus qu'à clôturer le merge via un commit

pour résoudre le conflit:

- 1- éditer les fichiers impactées
- 2- faire le merge des modifications
- 3- utiliser la commande "git add" suivi du nom du fichier pour indiquer à git que la résolution est terminé.
- 4- faire un commit: git commit

Info: il est possible d'annuler un merge en cours via la commande: git merge --abort

Faire un rebase d'une branche



le rebase d'une branche consiste à déplacer le point de départ de la branche vers un autre commit:

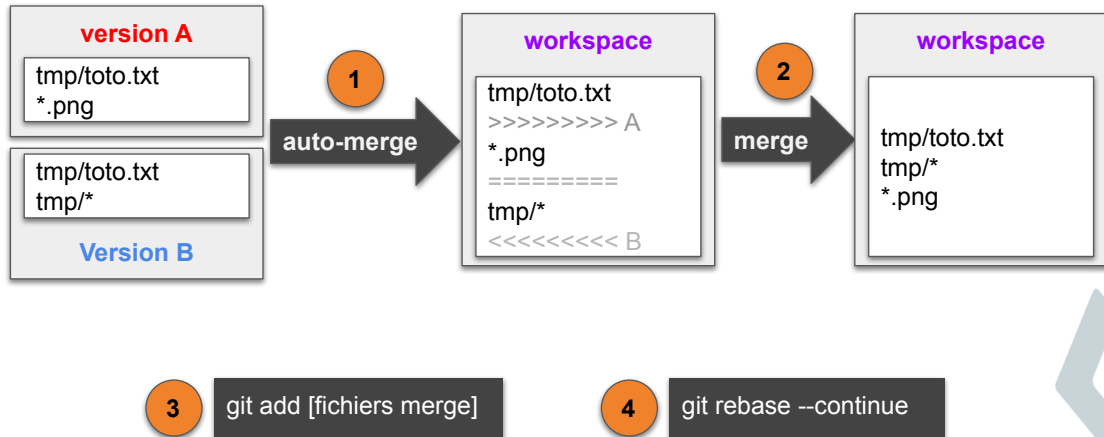
Attention: pour ne pas vous tromper de sens dans le rebase, il faut toujours activer la branche à modifier, puis la rebase sur l'autre. Par exemple, pour rebase la branche de travail develop sur la branche principale master, il faut:

- `switch develop`
- `rebase master`

Exemple (DEPOT):

- `gitk -all`
- `git switch develop`
- `git rebase master`
- `gitk -all`

Résoudre un conflit de rebase



Le rebase d'une branche peut entraîner des conflits lors du rejeu des commits.

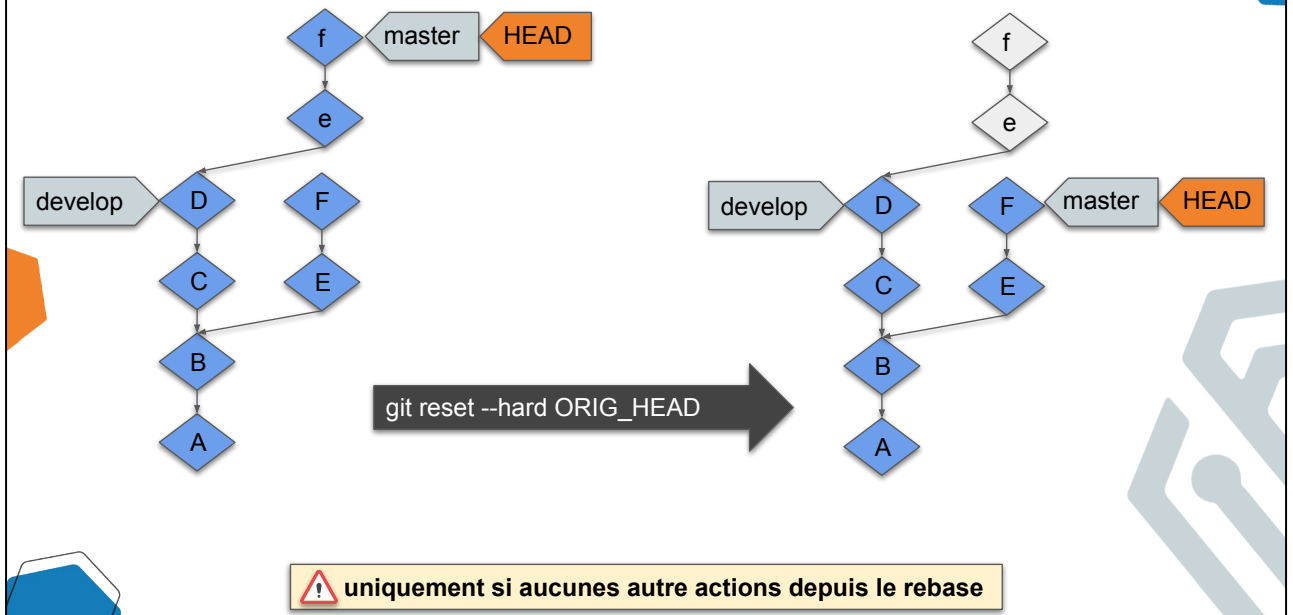
pour résoudre le conflit:

- 1- éditer les fichiers impactées
- 2- faire le merge des modifications
- 3- utiliser la commande "git add" suivi du nom du fichier pour indiquer à git que la résolution est terminée.
- 4- faire un git rebase --continue, pour passer au commit suivant

Info: il est possible d'annuler un rebase en cours via la commande: `git rebase --abort`

Attention: contrairement au merge, il ne faut pas faire de commit

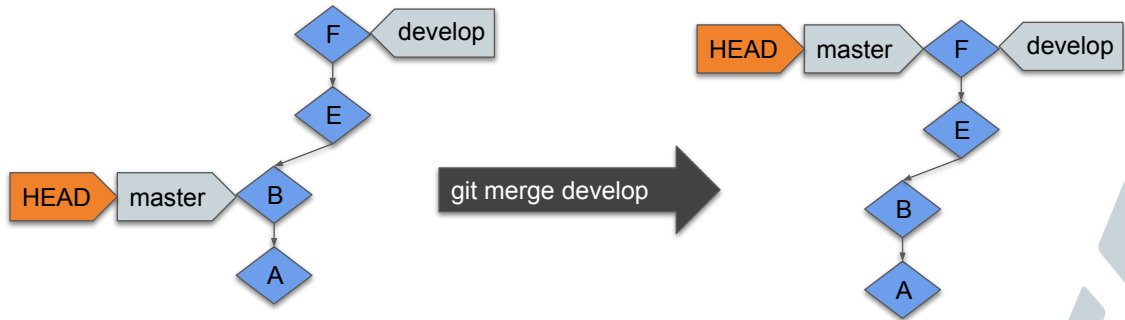
Annuler un rebase



Oups, j'ai fait mon rebase à l'envers: exemple, j'ai rebase master sur develop au lieu de l'inverse et pas de chance ça a fonctionné.

il est possible d'annuler l'action du rebase avec la commande `"git reset --hard ORIG_HEAD"` (plus d'info au chapitre sur le Reset)

Merge une branche rebase



Quand votre branche est rebase, cela ne signifie pas que la branche de destination comporte les modifications de la branche source. mais plutôt que votre branche source part maintenant du dernier commit de votre branche destination.

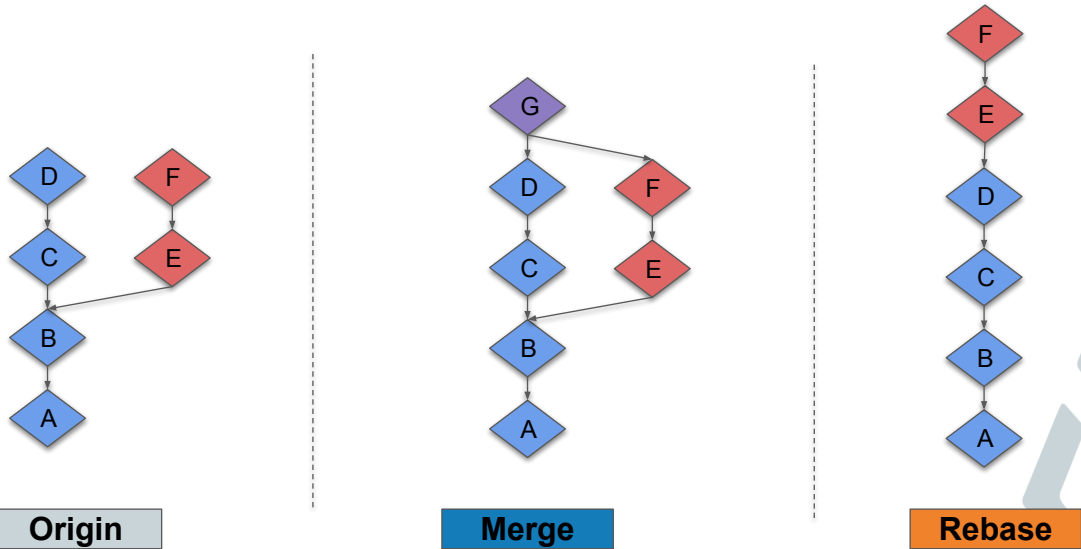
Il vous reste alors à merger la branche destination sur la source pour les ramener toutes deux sur le dernier commit:

- 1- `git switch <destination_rebase>`
- 2- `git merge <branche_a_rebase>`

Exemple (DEPOT):

- `gitk --all`
- `git switch master`
- `git merge develop`
- `git branch -d develop`
- `gitk --all`

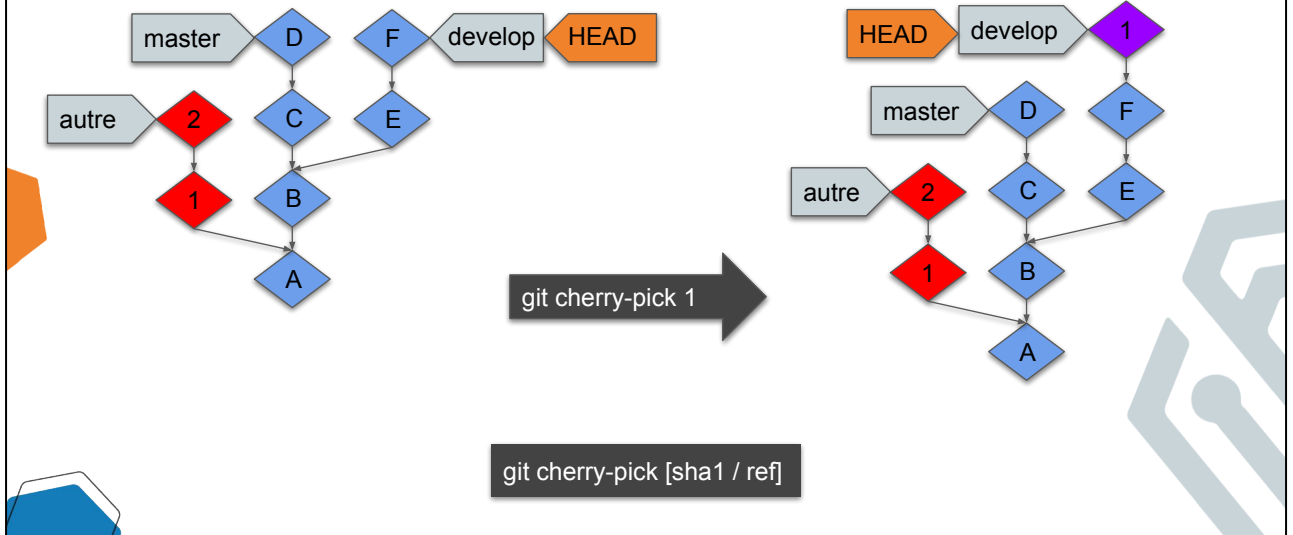
Merge Vs Rebase



Quand choisir le rebase ou le merge? et bien tout vas dépendre de ce que tu veux faire ressortir dans ton historique. On utilise plutôt le rebase si on souhaite avoir un historique linéaire. c'est le cas si les développements de ta branche aurais put être réalisé dans le master directement. on rebase et on supprime la branche de développement.

En revanche si tu souhaite conserver la branche de développement et marquer le fait que tu a réaliser ton développement dans une branche puis que tu a reversé les modifications dans la master, alors il faut utiliser le merge. Cela implique alors que ta branche de développement sera toujours présente dans ton historique, contrairement au rebase qui te permet de la faire disparaître

Cherry pick



le cherry-pick est une commande très pratique qui permet d'appliquer un commit dans une branche différente de celle où il se trouve:

- 1- se positionner sur la branche de destination
- 2- utiliser la commande "git cherry-pick" suivi du sha1 ou du tag du commit ciblé

cela peut être très pratique par exemple si une correction de bug est réalisée dans une branche et qu'une autre branche souhaite en profiter sans pouvoir ou vouloir les merge/rebase

TP 6.1

Scénario

Vous devez développer une fonctionnalité pour le produit sans perturber les corrections de bugs en cours sur le master.

Objectifs

- 1- Créer une branche de développement du nom de LIENS_EXT à partir du commit courant.
- 2- Faire le commit, sur la branche LIENS_EXT, de vos modifications en cours.
- 3- Ajouter un lien dans hello.html vers github
- 4- Faire un commit des modifications précédentes
- 5- Réaliser un **merge** de votre branche sur la branche master.
- 6- Supprimer la branche LIENS_EXT

SOLUTIONS:

1- il faut créer la branch et l'activer:

- `git switch -c LIEN_EXT`

2- `git commit -am "ajout liens"`

3- Ajout de la ligne "` Github`" dans le fichier hello.html:

4- `git commit -am "ajout github"`

5- Il faut activer master car c'est elle qui va changer

- `git switch master`
- `git merge LIEN_EXT`

6- `git branch -d LIENS_EXT`

TP 6.2

Scénario

Après annulation du merge du TP 6.1, vous vous retrouvez de nouveau sur la branche LIENS_EXT

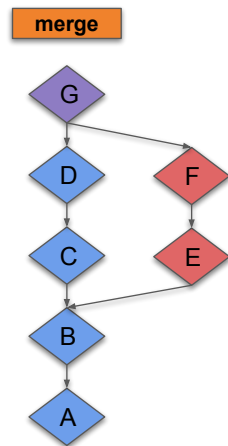
Objectifs

- 1- Réaliser un **rebase** de votre branche sur la master.
- 2- Faire le merge de la branche LIENS_EXT dans la branche master.
- 3- Supprimer la branche LIENS_EXT

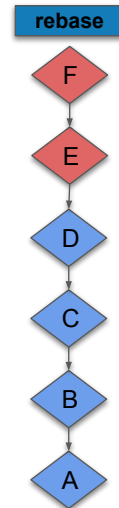
SOLUTIONS:

- 1- Il faut aller sur LIENS_EXT car c'est elle que l'on va rebase et donc modifier
 - `git switch LIEN_EXT`
 - `git rebase master`
 - Résolution des conflits
 - `git add .`
 - `git rebase --continue`
- 2- Il faut se déplacer sur master et faire le merge de LIENS_EXT:
 - `git switch master`
 - `git merge LIENS_EXT`
- 3- `git branch -d LIENS_EXT`

Bilan



```
git switch [destination_branch]  
git merge [branche_a_merge]
```



```
git switch [branche_a_rebase]  
git rebase [destination_rebase]
```

Nous avons vu le système de branche avec git et comment manipuler ces dernières.
Nous avons notamment vu comment merge ou rebase 2 branches entre elles