

# Les commandes merge et rebase

<b>Auteur</b> : Romain NOEL		<b>Date</b> : 02/03/2020	<b>Version</b> : 1
<b>Périmètre de l'opération</b> : Apprendre à fusionner les branches et à comprendre les différents types de fusion			
<b>Prérequis</b> : Poste de travail configuré			
<b>Outil</b> : Git		<b>Durée</b> : 30min	
<b>Opérations</b> (ce qu'il faut faire)	<b>Points clés</b> (à quoi il faut faire attention)	<b>Argumentaire</b> (pourquoi c'est important)	
Partie 1 : Fast Foward Merging			
<b>1/ Création d'un tag</b>  git checkout master git tag start		➤ Création d'un tag pour revenir plus facilement à cet endroit pour la suite de l'exercice	
<b>2/ Création d'une nouvelle branche</b>  git checkout -b fastfoward git log	Une nouvelle branche est créée, la HEAD a été déplacée dessus.		
<b>3/ Créer quelques commits</b>  touch fastfoward1 git add -a git commit -m "Premier commit pour fastfoward" touch fastfoward2 git add -a git commit -m "Second commit pour fastfoward" git log	La branche (et la HEAD qui lui est attachée) se déplace au fur et à mesure des commits	➤ La branche fastfoward est en avance de 2 commits sur la branche master. Il existe un chemin linéaire qui relie l'extrémité des deux branches.	
<b>4/ Faire un merge des deux branches</b>  git checkout master git merge fastfoward git log	Git vous indique qu'il effecture un fast foward merge.  Il n'y a pas eu de commit superflu de merge. La branche master a simplement été déplacée.	➤ Nous avons appliqué à la branche master les commits de la branche fastfoward.	

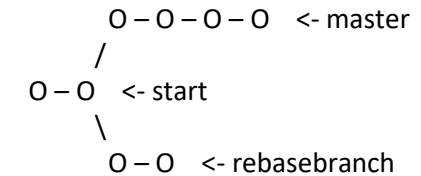
<b>5/ Supprimer la branche</b>  <pre>git branch -d fastforward git log</pre>	Aucun commit n'a été supprimé.	➤ La branche ne nous est plus utile, cette commande permet de la supprimer.
<b>Partie 2 : 3-way merging</b>		
<b>1/ Revenir au premier commit</b>  <pre>git checkout start</pre>	La HEAD est en retard de 2 commits par rapport à la branche master  <i>NB : Vous êtes en detached HEAD</i>	
<b>2/ Créer une branche</b>  <pre>git checkout -b threeway</pre>	<i>NB : Vous n'êtes plus en detached HEAD</i>	
<b>3/ Création d'un nouveau commit</b>  <pre>touch 3way1 git add -a git commit -m "Premier commit pour 3way" touch 3way2 git add -a git commit -m "Second commit pour 3way" git log</pre>	La branche threeway se déplace avec la HEAD.	➤ La branche threeway diverge de la branche de manière non-linéaire :  <pre>       O - O  &lt;- master       / O - O  &lt;- start       \       O - O  &lt;- threeway </pre>
<b>4/ Faire un merge des deux branches</b>  <pre>git checkout master git merge threeway</pre> Git vous ouvre une console vi pour que vous y inscriviez un message de commit <pre>git log</pre>	Git crée un commit qui est la fusion des deux branches.  Tous les commits sont visibles à travers le git log	➤ Les deux branches sont fusionnées et il y a un commit supplémentaire qui est un commit purement technique résultant de la fusion des deux branches.
<b>5/ Supprimer la branche</b>  <pre>git branch -d threeway git log</pre>		➤ Suppression de la branche qui ne nous est plus utile.

## Partie 3 : Le rebase

### 1/ Créer une branche divergente de la branche principale

```
git checkout start
git checkout -b rebasebranch
touch rebase1
git add -a
git commit -m "Premier commit pour le rebase"
touch rebase2
git add -a
git commit -m "Second commit pour le rebase"
git log
```

- La branche threeway diverge de la branche de manière non-linéaire :



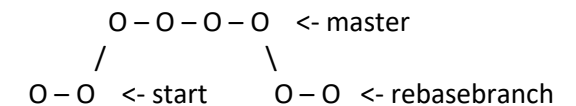
### 2/ Création d'un nouveau commit

```
git rebase master
ls
git log
```

Il n'a pas été créé de commit supplémentaire lors du rebase.

Tous les commits de la branche master ont été appliqués. Vous travaillez avec vos commits plus ceux de la branche master.

- La commande rebase déplace tous les commits de la branche et les attache à la branche cible :



### 3/ Merge et suppression de la branche rebase

```
git checkout master
git merge rebasebranch
git branch -d rebasebranch
```

Il s'agit d'un fast forward merge

- On applique les commits de la branche rebase à la branche master

