Les commandes merge et rebase

Auteur : Romain NOEL	**Date :** 02/03/2020	**Version :** 1
Périmètre de l'opération : Apprendre à fusionner les branches et à comprendre les différents types de fusion		
Prérequis : Poste de travail configuré		
Outil : Git	**Durée :** 30min	

Opérations (ce qu'il faut faire)	**Points clés** (à quoi il faut faire attention)	**Argumentaire** (pourquoi c'est important)
Partie 1 : Fast Foward Merging		

1/ Création d'un tag
 	
`git checkout master`
`git tag start`
		➢ Création d'un tag pour revenir plus facilement à cet endroit pour la suite de l'exercice

2/ Création d'une nouvelle branche
 	
`git checkout -b fastfoward`
`git log`
	Une nouvelle branche est créée, la HEAD a été déplacée dessus.	

3/ Créer quelques commits
 	
`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.

4/ Faire un merge des deux branches
 	
`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.

5/ Supprimer la branche
 	
`git branch -d fastfoward`
`git log`
	Aucun commit n'a été supprimé.	➢ La branche ne nous est plus utile, cette commande permet de la supprimer.

Partie 2 : 3-way merging		
1/ Revenir au premier commit `git checkout start`	La HEAD est en retard de 2 commits par rapport à la branche master *NB : Vous êtes en detached HEAD*	
2/ Créer une branche `git checkout -b threeway`	*NB : Vous n'êtes plus en detached HEAD*	

3/ Création d'un nouveau commit
 	
`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`
	La branche threeway se déplace avec la HEAD.	➢ La branche threeway diverge de la branche de manière non-linéaire :
		
		O – O <- master
		/
		O – O <- start
		\
		O – O <- threeway

4/ Faire un merge des deux branches
 	
`git checkout master`
`git merge threeway`
Git vous ouvre une console vi pour que vous y inscriviez un message de commit
`git log`
	Git créée 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.

5/ Supprimer la branche
 	
`git branch -d threeway`
`git log`
		➢ 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 : O – O – O – O <- master / O – O <- start \ O – O <- rebasebranch
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 : O – O – O – O <- master / \ O – O <- start O – O <- rebasebranch
3/ Merge et suppression de la branche rebase git checkout master git merge rebasebranch git branch -d rebasebranch	Il s'agit d'un fast foward merge	➢ On applique les commits de la branche rebase à la branche master O – O – O – O / \ O – O <- start O – O <- master <- ~~rebasebranch~~