Les commandes checkout, tag, branch

Auteur : Romain NOEL	**Date :** 02/03/2020	**Version :** 1
Périmètre de l'opération : Gérer un mode de développement non-linéaire		
Prérequis : Poste de travail configuré		
Outil : Git	**Durée :** 45min	

Opérations (ce qu'il faut faire)	**Points clés** (à quoi il faut faire attention)	**Argumentaire** (pourquoi c'est important)
Partie 1 : Se déplacer dans son historique		
1/ Création d'un tag git tag start		➢ Création d'un tag qui correspond à l'état du code au départ de l'exercice.
2/ Création d'un nouveau commit touch exo5.txt git add exo5.txt git commit -m "Création du jeu de test de l'exercice 5"	Indexation d'un seul fichier. Seul fichier1.txt doit être pris en compte lors de la sauvegarde.	➢ Création du jeu de données pour l'exercice
3/ Regarder la list des différents commits, tags et branches git log	La HEAD pointe sur la branche master, qui pointe sur le dernier commit. Le premier commit est taggé « start ».	➢ La commande log affiche également la position de la HEAD (commit courant), ainsi que la relation entre les commits et les éventuels noms (tag ou branche)
4/ Afficher la liste des tags git tag --list	Il n'existe pas de tag par défaut, le seul tag que vous voyez est celui que vous avez créé.	➢ Cette commande permet de lister les tags sans autre forme d'information
5/ Se déplacer dans l'historique via le #commit ls git log <copier l'ID du premier commit> git checkout ***#commit*** ls git log	Un #commit est long et difficile à mémoriser. Il ne contient aucune information qui indique ce qu'il représente. Le fichier exo5.txt a disparu, il s'agit de l'état de votre code avant la création de ce fichier. *NB : Vous êtes en position de « detached head », la head ne pointe sur aucune branche.*	➢ Cette commande permet de se déplacer dans l'historique (à condition de connaître les numéros de commits) ➢ La commande git log n'indique que les commits précédents

6/ Se déplacer via le nom de branche git checkout master ls git log	Un nom de branche est plus facile à mémoriser. Son nom doit représenter la raison pour laquelle elle a été créée. *NB : Vous n'êtes plus en detached HEAD*	
7/ Se déplacer grâce au tag git checkout start ls git log	Un tag est fortement lié à un commit. Il ne peut (et ne doit) pas être modifié une fois créé (contrairement aux branches qui vont évoluer avec les commits).	➢ Cette commande est équivalente au déplacement par #commit
8/ Revenir sur le master git checkout master		
9/ Supprimer le tag git tag -d start		➢ Cette commande supprime le tag (sans toucher au commit)
Partie 2 : Créer une branche		
1/ Créer une branche git checkout -b experimentation git log	Une nouvelle branche « expérimentation » a été créée Votre HEAD est maintenant attachée à la nouvelle branche. Elle n'est plus attachée à la branche master.	
2/ Lister les branches git branch -a	Marque d'un « * » la branche sur laquelle vous vous trouvez.	➢ Permet de lister toutes les branches (locales et distantes) ➢ Vous indique la branche sur laquelle vous vous trouvez (information également disponible avec la commande git log)
3/ Création d'un nouveau commit touch experimentation.txt git add experimentation.txt git commit -m "Expérimentation pour l'exercice 5" git log	La branche master est « restée » sur le commit précédent. La branche experimentation s'est déplacée sur le nouveau commit. La HEAD pointe sur la branche experimentation (et donc sur le dernier commit)	➢ La HEAD se déplace avec la branche sur laquelle elle pointe ➢ La branche se déplace sur le dernier commit

4/ Faire évoluer la branche master en parallèle git checkout master touch master.txt git add master.txt git commit -m "Evolution de master pour l'exercice 5" ls git log	Le fichier experimentation n'est pas présent La branche experimentation n'apparaît pas dans le git log. La HEAD est attachée au master	➢ Le seul moyen de retrouver le nom de la branche pour s'y déplacer est la commande git branch -a
Partie 3 : Le Detached HEAD		
1/ Se mettre en état de detached head git log <copier l'ID du commit précédent> git checkout ***#commit*** git log	Vous êtes en état de « detached head »	➢ La HEAD ne pointe sur aucune branche.
2/ Création d'un nouveau commit touch detached.txt git add detached.txt git commit -m "Création d'un commit en detached HEAD"	Git vous rappelle que vous êtes en detached HEAD	➢ Vous venez de créer un commit qui diverge des différentes branches et qui n'appartient à aucune autre branche.
3/ Retour sur le master git checkout master git log	Git vous notifie qu'un de vos commit n'est connecté à aucune branche. Il vous propose d'ailleurs d'en créer une afin de ne pas perdre votre commit. Votre commit n'est plus visible dans les logs. A moins d'avoir noté le #commit quelque part, vous n'avez plus de moyen de le retrouver.	➢ Il n'existe pas de commande pour retrouver votre commit. Ces travaux sont perdus.