Les commandes commit, reset et log

Auteur : Romain NOEL	**Date : 02/03/2020**	**Version : 1**
Périmètre de l'opération : Créer des sauvegardes en local, en supprimer et vérifier l'état courant de son dépôt		
Prérequis : Poste local configuré		
Outil : Git	**Durée : 15**min	

Opérations (ce qu'il faut faire)	**Points clés** (à quoi il faut faire attention)	**Argumentaire** (pourquoi c'est important)

1/ Créer deux fichiers
 	
`touch fichier1.txt`
`touch fichier2.txt`
	La commande touch permet de créer un fichier vide.	➢ Création du jeu de données pour l'exercice

2/ Indexer un fichier
 	
`git add fichier1.txt`
`git status`
	Indexation d'un seul fichier. Seul fichier1.txt doit être pris en compte lors de la sauvegarde.	➢ Les différentes modifications peuvent être séparées pour faire partie de différentes sauvegardes.
		➢ Ajoute les fichiers à l'index pour que le fichier soit pris dans le prochain commit.

3/ Créer une sauvegarde `git commit -m "Création de fichier1"`	Git n'affiche pas de message de confirmation. C'est normal.	➢ Cette commande permet d'intégrer dans le dépôt local tous les fichiers indexés. ➢ Toutes les modifications du fichier ont été commitées. Elles sont dans le dépôt local.
4/ Afficher la liste et les informations des commits `git log`		➢ Cette commande permet de connaitre les informations du commit : ID, auteur, date, commentaire, etc… ➢ Utiliser l'ID de commit comme identifiant unique. Les 7 premiers chiffres devraient être suffisant pour l'identifier.

5/ Créer la sauvegarde du second fichier
 	
`git add fichier2.txt`
`git status`
`git commit -m "Création de fichier2"`
`git log`
		➢ Création d'une seconde sauvegarde

6/ Annuler la dernière sauvegarde
 	
`git reset HEAD^`
`git log`
	La commande git-reset est pertinente tant que le commit n'a pas été pushé.	➢ Annule le commit sur le dépôt local et revient à l'avant dernier commit. Les modifications dans le fichier ont été supprimées.
		➢ La commande reset retire le commit dans git