LEXIQUE

Dépôt / Repo(sitory)
Base de données généralement stockée dans le dossier .git à la racine du working directory. Elle contient l'ensemble des objets (commits, blobs, etc.) et des références (branches, tags) nécessaires à la gestion des sources d'un projet.

Remotes
Références vers des dépôts distants, servant de points d'échange avec les autres collaborateurs. Ils permettent de synchroniser le travail entre plusieurs contributeurs.

Snapshot
Capture instantanée d'un état de fichiers versionnés. Contrairement à d'autres systèmes de contrôle de version, Git enregistre des snapshots complets des fichiers, plutôt que des diffs (différences).

SHA1 (« Chawan »)
Identifiant unique calculé via un algorithme de hachage, utilisé pour identifier les objets Git (commits, tags, blobs, etc.). Ce hash est composé de 40 caractères hexadécimaux, garantissant son unicité.

Commit
Unité fondamentale en Git, correspondant à un snapshot du code source au moment où l'on souhaite enregistrer une version. Chaque commit contient un SHA1, des métadonnées (auteur, date, message), et pointe vers son ou ses parents (commits précédents).

WD / Working Directory
Dossier contenant les fichiers d'un projet géré par Git. À sa racine se trouve généralement le dossier .git. Un dépôt "nu" (bare repository) n'a pas de working directory associé.

Stage / Index
Zone intermédiaire entre le working directory et le dépôt local. Elle permet de préparer les modifications avant de les valider via un commit. Les fichiers sont ajoutés à cette zone avec la commande git add.

Stash
Zone temporaire et locale au dépôt, utilisée pour mettre de côté des modifications en cours (non encore commités ou non ajoutés au stage). Cela permet de revenir à un état propre et de reprendre ces travaux plus tard.

Historique / log
Affichage de la séquence des commits qui composent l'évolution d'un dépôt. Par défaut, il est présenté en ordre chronologique inverse. Étant donné la présence de branches et de fusions, cet historique forme un graphe plutôt qu'une ligne continue.

Tag
Étiquette associée à un commit pour le rendre plus facilement identifiable. Contrairement aux branches, les tags sont fixes et ne changent pas de référence une fois définis.

HEAD
Pointeur représentant le commit actuellement actif dans le dépôt. Il sert de référence pour comparer les modifications entre le stage et le working directory.

ORIG_HEAD
Pointeur temporaire sur le commit référencé précédemment par HEAD, souvent utilisé pour revenir à un état antérieur.

Reflog
Historique strictement local et privé des positions successives du HEAD dans le dépôt. Il permet de retrouver un état précédent même si celui-ci n'est plus accessible via les branches ou les tags.

Tête détachée (detached head)
État où le HEAD pointe directement vers un SHA1 au lieu de référencer une branche. Dans cet état, les commits effectués ne seront pas rattachés à une branche active et risquent d'être perdus sans intervention.

Reset
Commande permettant de déplacer explicitement le HEAD et la branche courante sur un commit donné, avec possibilité de modifier ou conserver les modifications non commités.

Fusion / Merge
Action d'intégrer les modifications d'une branche dans une autre. Git tente automatiquement de fusionner les fichiers modifiés, mais peut nécessiter une intervention manuelle en cas de conflits.

Conflit
Situation où Git ne peut pas résoudre automatiquement les différences entre deux modifications sur un même fichier lors d'une fusion, d'un rebase ou d'autres opérations similaires. L'utilisateur doit alors arbitrer manuellement.

Rebase
Action de réécrire l'historique d'un commit, généralement une branche entière, soit en la déplaçant (en changeant son point de base : re-base), soit en réécrivant simplement son historique propre (pour le nettoyer), soit les deux.

INSTALLATION

Outil Git + interphases graphique (git et gitk).

Linux: sudo apt-install git-all
Mac/Windows: https://git-scm.com/download

COMMANDES

CONFIGURATION

Configurer git pour le dépôt courant ou pour tous avec l'option --global

$ git config --global user.name "[nom]"
Définit le nom de l'utilisateur

$ git config --global user.email [email]
Définit l'email de l'utilisateur

GESTION DES DÉPÔTS

Créer un dépôt local ou en utiliser un distant via son url.

$ git init --bare [nom_dépôt]
Créer un dépôt local vide dans le dossier courant ou en créer un avec le nom spécifié. **--bare** permet de créer un dépôt sans workspace.

$ git clone [url] [nom_dossier]
Duplique en local le dépôt git pointé par l'url. Le dépôt local comportera alors un remote du nom de origin.

$ git remote -vv
Liste les remotes du dépôt courant.

$ git remote add [remote_name] [remote_url]
Ajoute un remote au dépôt courant.

$ git remote rm [remote_name]
Supprime le remote du dépôt courant.

$ git remote set-url [remote_name] [new_url]
Modifier l'url de la remote ciblée.

SYNCHRONISER LES CHANGEMENTS

Synchroniser le dépôt local et distant.

$ git fetch [remote]
Met à jour le dépôt locales avec les informations du serveur distant mais ne les merge pas avec la branche courante et le workspace.

$ git pull [remote] [branch]
Met à jour le dépôt locales avec les informations du serveur distant et les merge avec la branche courante et le workspace.

$ git push -d [remote] [branch/Tag]
Synchronise le serveur distant et le remote. Permet d'envoyer les modifications de la référence locale vers le remote. l'option **-d** permet de supprimer la référence sur le serveur.

GÉRER LES MODIFICATIONS

Voir les changements, les sélectionner et les enregistrer.

$ git status
Donne l'état courant du workspace et dépôt local.

$ git (checkout ou restore) [fichier]
Supprime les modifications courantes du fichier. Attention, impossible de les récupérer après

$ git add --patch [fichiers]
Ajoute les modifications d'un fichier a la zone d'index. Mode interactif avec --patch pour une indexation sélective.

$ git (reset HEAD ou restore --staged) [fichiers]
Enlève, de la zone d'index, les modifications mais les conserves dans le workspace.

$ git restore [fichiers]
On applique dans le WD la version de l'index du fichier

$ git checkout [fichier]
Annule la modification du fichier dans le WD

$ git checkout [tag|sha1|branche] [fichier]
Applique dans la zone d'index la version du fichier correspondant à la référence donnée.

$ git diff --staged [fichier]
Montre les modifications du fichier non indexé, pour un fichier indexé, ajouter l'option **--staged**.

$ git commit --amend -m "[message]"
Enregistre les modifications présentes dans la zone d'index dans l'historique du dépôt. **--amend** permet de 'modifier' le commit précédent.

$ git rm --cached [fichier]
Supprime du suivi de git le fichier. Ce fichier est également supprimé du WD sauf avec --cached

$ git mv [src] [dest]
Déplace/Renomme un fichier dans le WD tout en conservant le suivi de son historique.

GESTION DE L'HISTORIQUE

Visualiser et naviguer dans l'historique du dépôt Git.

$ **git log [sha1|ref]** **--oneline --graph --all**
Affiche l'historique des commits depuis la position de la ref ou sha1 (sinon HEAD).
--oneline: pour limiter à une ligne par commit.
--graph: pour afficher sous forme d'arbre.
--all: pour afficher toutes les références.

$ git diff [tag|sha1] [tag|sha1]
Affiche la différence de contenu entre deux commits.

$ git show [tag|sha1]
Affiche les modifications apportées par un commit.

$ git checkout [tag|sha1|branche]
Se déplacer dans l'historique sur le commit donné

$ git tag **-n**
Liste les tags présents dans le dépôt local. **-n** affiche également les messages associés

$ git tag **-d** **[nom_tag]**
Créer un tag sur le commit courant. L'option **-d** permet de supprimer le tag.

RÉÉCRIRE L'HISTORIQUE

Corriger des erreurs, regrouper ou modifier des commits.

$ git revert [tag|sha1|branche]
Créer un nouveau commit qui est l'opposé du commit spécifié afin d'annuler ses effets.

$ git reset **[--soft|--mixed|--hard]** **[tag|sha1|branch]**
Déplace HEAD et la tête de la branche courante. l'option par défaut est **--mixed**
-- soft: conserve le workspace et la zone d'index
-- mixed: conserve le workspace uniquement
-- hard: ne conserve pas le workspace et l'index

$ git rebase –i [tag|sha1|branch]
Réordonne, fusionne, supprime, modifie les n dernier commits local. A ne pas faire sur des commit poussé sur le remote.

TRAVAILLER EN BRANCHE

Créer et gérer les branches en local et via les remotes

$ git switch [branche]
Changer de branche courante.

$ git branch -u [remote/branch] [branch]
Fait suivre une branche distante par une branche locale.

$ git branch **-r** **-a**
Liste toutes les branches locales dans le dépôt courant. **-r** pour les branche distantes. **-a** pour tout

$ git branch **-d** **[nom_branche]**
Créer une nouvelle branche. **-d** pour supprimer la branche local.

$ git (checkout **-b** **ou switch** **-c**) **[nom_branche]**
Change de branche courante. **-b ou -c** pour en plus créer la branche.

$ git merge [autre_branche]
Combine dans la branche courante l'historique de la branche spécifiée via un commit de merge.

$ git rebase **-i** **[autre_branche]**
Déplace les commits de la branche courante sur la branche spécifiée. **-i** Pour le mode interactif

$ git cherry-pick [commit]
Appliquer un commit à l'espace de travail.

METTRE DE CÔTÉ DES MODIFICATIONS

Mètre de côté des modifications temporairement.

$ git stash save "[message]"
Enregistre toutes les modifications courante dans une pile temporairement.

$ git stash list
Liste toutes modifications mises de côté.

$ git stash pop id
Appliquer les modifications à l'index id de la pile dans l'espace de travail. sinon id = 0

$ git stash drop id
Supprime les modifications à l'index id de la pile. sinon id = 0.

DEBUGER SON CODE

Outils d'aide au debug.

$ git blame -L [line_start],[line_stop] [file_name]
Afficher qui et quand chaque lignes du fichier ont été modifiés.

$ git bisect [commit_bad] [commit_good]
Recherche par dichotomie l'origine d'un bug entre deux commits.

GIT SUBMODULE

$ git clone --recurse-submodules [repo-url]
Cloner un dépôt et ses sous modules

$ git submodule add [repo-url] [local-path]
Ajoute un sous module au dépôt courant

$ git submodule update
Mettre à jour les sous modules

$ git push --recurse-submodules=check
Pousse le dépôt principale et vérifie que les sous modules sont également synchronisés avec le remote

$ git submodule foreach [cmd]
Applique la commande aux sous modules

GIT LFS

$ git lfs install
Installer le module git LSF

$ git lfs track [regex_file]
Ajoute un type de fichier a la gestion de LFS

Arnaud MERCIER
www.hexotech.fr

HEXOTECH