**Principales commandes GIT**

Initialiser un nouveau reposotory

git init

Afficher les changements en attente

git status

Ajouter en staging un fichier

git add <nom du fichier>

Ajouter en staging plusieurs fichiers en même temps

git add <nom fichier1> <nom fichier2>

Ajouter tous les changements en staging

git add .

git add --all

Retirer un fichier qui avait été ajouté pour être « commité »

git reset HEAD -- <fichier à supprimer>

Pour supprimer un fichier de GIT et le garder localement

git rm –-cached <nom du fichier>

Supprimer un fichier du répertoire de travail et mise à jour de l’index

git rm –f <nom du fichier>

Afficher les changements apportés à un fichier

git diff <nom du fichier>

Créer un commit (i.e., commiter les changements)

git commit –m "message du commit"

#### Modifier le dernier message de commit

git commit –m "nouveau message" --amend

Revenir à un commit précis

git reset <ID du commit>

Annuler le dernier commit

git reset HEAD^

Cela annule le dernier commit et revient à l’avant-dernier.

Pour indiquer à quel commit on souhaite revenir, il existe plusieurs notations :

* HEAD : dernier commit ;
* HEAD^ : avant-dernier commit ;
* HEAD^^: avant-avant-dernier commit ;
* HEAD~2 : avant-avant-dernier commit (notation équivalente) ;

Seul le commit est retiré de Git; les fichiers restent modifiés.

Annuler tous les changements du dernier commit (hard)

Si vous voulez annuler votre dernier commit **et** les changements effectués dans les fichiers, il faut faire un *reset hard*.

git reset --hard HEAD^

Tous les changements seront perdus. Faire attention avant d’utiliser cette commande.

git reset --soft HEAD^

Revient en arrière et laisse toutes les modifications en stagging.

git reset --mixed HEAD^

Revient en arrière, laisse les modifications, mais ne les « stage » pas.

Si vous avez modifié plusieurs fichiers mais que vous n’avez pas encore envoyé le commit et que vous voulez restaurer un fichier tel qu’il était au dernier commit.

git checkout <nom du fichier>

git checkout <ID du commit> permet de voir un commit particulier tel qu’il était sans perdre les modifications faites depuis. Les modifications qui seront apportées ne seront pas sauvegardées.

Retirer un fichier qui avait été ajouté pour être « commité »

git reset HEAD -- <fichier à supprimer>

Afficher l’historique des commit

git log

Afficher une seule ligne par commit

git log –oneline

Afficher seulement les derniers commit

git log –n <nombre de commit à afficher>

Pour observer l’historique d’un fichier particulier

git log –-oneline –p <nom du fichier>

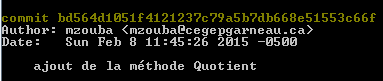
On peut aussi utiliser gitk pour afficher l’historique

gitk

Annuler un commit (défaire un commit)

git revert <ID du commit>

Il n’est pas nécessaire d’indiquer l’ID du commit en entier. Les 5-6 premiers chiffres peuvent suffire (s’il n’y a pas d’ambigüité).



**Exemple :** Pour annuler le commit « ajout de la méthode Quotient », on pourrait faire :

git revert bd564d

Afficher les branches

git branch

La branche courante est marquée d’un astérisque

Créer une nouvelle branche à partir de la branche courante

git branch <nom de la branche>

Changer de branche

git checkout <nom de la branche>

Créer une nouvelle branche et se positionner sur cette nouvelle branche

git checkout –b <nom de la branche>

Fusionner une branche à la branche courante

git merge <nom de la branche>

Supprimer une branche

git branch -d <nom de la branche>

Git vérifie que votre travail dans la branche a bien été fusionné dans « master ». Sinon, il vous en avertit et vous interdit de supprimer la branche (vous risqueriez sinon de perdre tout votre travail dans cette branche !).

Pour supprimer une branche même si elle contient des changements que vous n’avez pas fusionnés utilisez l’option -D (lettre capitale) :

git branch -D <nom de la branche>

#### Mettre de côté le travail en cours avant de changer de branche

#### Pour éviter d’avoir à faire un commit au milieu d’un travail en cours, on écrit :

git stash

Les fichiers modifiés seront sauvegardés et mis de côté. On peut alors changer de branche, faire des modifications, « committer », puis revenir sur la branche où on était. Pour récupérer les changements mis de côté, on écrit :

git stash apply

Pour afficher les stash

git stash list

Pour effacer les stash

git stash drop

Pour ajouter un tag sur un commit :

git tag <nom du TAG> <ID du commit>

**Exemple**

git tag v1.3 2f7c8b3428aca535fdc970feeb4c09efa33d809e

Pour que le tag soit envoyé lors d’un push, il faudrait ajouter l’option - - tag :

git push --tags

Pour supprimer un tag

git tag -d <nom du TAG>

Pour télécharger un projet et tout son historique

git clone <url du projet>

Exemple : Pour cloner le projet Symfony2 à partir de Github, on écrirait :

git clone http://github.com/symfony/symfony.git

Cela va créer un dossier « symfony » et y télécharger tous les fichiers source du projet ainsi que l’historique de chacune de leurs modifications.

Pour envoyer les changements à votre dépôt distant, exécutez la commande

git push origin master

Remplacez master par la branche dans laquelle vous souhaitez envoyer vos changements.

Si vous n'avez pas cloné votre dépôt existant et voulez le connecter à votre dépôt sur un serveur distant, vous devez l'ajouter avec

git remote add origin <url du dépôt>

Maintenant, vous pouvez envoyer vos changements vers le serveur distant sélectionné.

Pour récupérer et fusionner les changements distants dans votre espace de travail, exécutez la commande

git pull origin master

Pour récupérer les fichiers sans les fusionner, exécuter la commande

git fetch origin master

**Références**

Livre ProGit en français

<http://git-scm.com/book/fr/v1>

Gérez vos codes sources avec Git

<http://openclassrooms.com/courses/gerez-vos-codes-source-avec-git>

Maitrisez Git en 28 minutes

<https://www.youtube.com/watch?v=5IcYILdejs8>

Comprendre Git

<https://www.youtube.com/watch?v=rP3T0Ee6pLU&index=1&list=PLjwdMgw5TTLXuY5i7RW0QqGdW0NZntqiP>

Modèle de branche avec Git

<http://www.occitech.fr/blog/2014/12/un-modele-de-branches-git-efficace/>

Git

<https://gist.github.com/aquelito/8596717>

<http://www.miximum.fr/enfin-comprendre-git.html>

Git- Petit guide

<http://rogerdudler.github.io/git-guide/index.fr.html>

10 commandes utiles

<http://pioupioum.fr/developpement/git-10-commandes-utiles.html>