**Commandes en Git**

Voc :

repository = dépôt

commit = version

stage = index répertoriant tous les éléments modifiés à envoyer lors de la prochaine sauvegarde

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Symbole | Signification | Code | | Utilisation | |
| Commandes générales Linux |  |  | |  | |
| pwd | indique le répertoire courant | pwd | |  |  |
| ls | donne la liste des fichiers et dossiers dans le répertoire courant  sous forme de liste  en liste avec plus de détails | ls  ls -l  ls -l -a | |  |  |
| cat | affiche le contenu d'un fichier | cat monFichier.txt | |  |  |
| mkdir | créer un dossier | mkdir monDossier | | *créer un nouveau dossier et se placer à l'intérieur*  mkdir monDossier && cd $\_ |  |
| touch | créer un fichier | touch monFicier.txt | |  |  |
| cd | se placer dans le dossier indiqué  remonte l'arborescence de 2 dossiers  revient au répertoire principal | cd monDossier\monSousDossier  cd ..\..  cd ~ | |  |  |
|  |  |  | |  |  |
|  |  |  | |  |  |
|  |  |  | |  |  |
|  |  |  | |  |  |
|  |  |  | |  |  |
| Initialisation de Git | après avoir installé Git sur la machine, il faut configurer les éléments suivants : |  | |  |  |
| config | définir le nom ou pseudo et adresse mail qui serviront de ref dans les commits et à lier le compte GitHub | git config --global user.name 'Mon Nom'  git config --global user.email 'monmail@sfr.fr' | |  |  |
|  |  |  | |  |  |
|  |  |  | |  |  |
| Commandes Git | ATTENTION : utilisation de GitBash pour pouvoir utiliser les commandes version Linux |  | |  |  |
| help | obtenir l'aide liée à la commande donnée en paramètre | git help maCommande | |  |  |
| init | initialise le repository dans le dossier dans lequel on est positionné | git init | |  |  |
| ls -l -a .git | passe en revue les éléments du dossier .git | ls -l -a .git | |  |  |
| cat | accède au contenu du fichier | cat monFichier.txt | |  |  |
|  |  |  | |  |  |
| Commit |  |  | |  |  |
| status | permet de connaitre l'état du repository (fichiers untracked, unmodified, modified, staged, …) |  | |  |  |
| add | ajoute le fichier ou le dossier au stage, cad aux éléments qui seront sauvegardés lors du prochain commit | git add monFichier | | *Ajout de tous les fichiers du dossier courant :*  git add . |  |
| rm--catched | retire le fichier/dossier du stage | git rm --cached monFichier | |  |  |
|  |  |  | |  |  |
| .gitignore | fichier texte à créer à la racine du dossier de travail et à remplir avec les titres des fichiers à ignorer | mkdir .gitignore  *remplissage manuel du fichier*  git add .gitignore | *Contenu possible d'un fichier* **.gitignore** *:*  # un commentaire  *ignore*  \*.a  *fichiers de type a*  !lib.a *sauf pour le fichier lib.a*  /TODO  *le fichier TODO à la racine du projet*  build/  *tout le répert. build*  doc/\*.txt fichiers .txt directement dans le  dossier doc  doc/\*\*/\*.txt tous les .txt dans le dossier et sous-  dossier de doc | | ATTENTION : .gitignore doit faire partie des fichiers suivit dans le stage et commité afin de transmettre les infos qu'il contient ! |
| commit -m | crée la version, cad sauvegarde toutes les fichiers contenus dans le stage et ayant étés modifiés | git commit -m ’Nom du commit' | |  |  |
| commit -a -m | crée le commit directement sans avoir à passer par le stage (cad git add …)  ATTENTION : n'inclus alors dans le commit uniquement les fichier déjà suivis dans les précédents commits (pas les nouveaux) | git commit -a -m’Nom du commit' | |  |  |
| log | affiche l'historique des commits, du plus récent au plus ancien | git log | |  |  |
| -p | montre les différences introduites entre chaque commit | git log -p | |  |  |
| -2 | limite le nombre de résultats au nombre indiqué en paramètre | git log -10 | |  |  |
| cat-file -p | accède au contenu d'un fichier référencé dans le commit avec sa hash de 40 ca. | git cat-file -p 25adf65… | |  |  |
| blame | accède aux auteurs qui ont modifié chaque ligne d'un fichier particulier et au début de la Hash du commit qui a validé la ligne | git blame monFichier.txt | |  |  |
| show | accède aux détails et changements dans un commit à partir du début de sa hash | git show 2221cd93 | |  |  |
| tag | affiche la liste des tags | git tag | |  |  |
| tag -a -m | place un tag (étiquette) sur le commit en cours | git tag -a MonTag -m ’message associé au tag' | |  |  |
| stash | zone mémoire permettant d'utiliser checkout sans perdre le travail en cours en sauvegardant temporairement les modifications des fichiers sans avoir à les commiter, le temps de traiter d'autres éléments sur un autre commit | git stash | |  |  |
|  | récupérer les fichiers  en vidant le stash  sans vider le stash | git stash pop  git stash apply | |  |  |
|  |  |  | |  |  |
|  |  |  | |  |  |
|  |  |  | |  |  |
|  |  |  | |  |  |
| commit -a --amend -m | modifie le dernier commit | git commit -a --amend -m’Nouveau nom du commit' | |  | ATTENTION : à éviter car il y a modification de l'historique, le commit précédent étant détruit pour être remplacé par le nouveau ! |
| revert | créé un nouveau commit qui annule les modifications du commit précédent (du commit parent) | git revert 25dghb5256sdg5s2… | |  |  |
| reset --hard | annule tous les changements des fichiers qui sont dans le stage, mais n'ont pas encore été commités | git reset --hard | |  |  |
|  |  |  | |  |  |
|  |  |  | |  |  |
| Branches |  |  | |  |  |
| checkout | changer de commit ou de branche (se place sur le dernier commit de la branche) | git checkout 25dhfz652dfkhdf…  git checkout maBranche | | git checkout master  git chechout HEAD^^  git checkout HEAD~2 | => retourne sur le dernier commit de master  => remonte de deux commits parents  => remonte de deux commits parents |
| branch | afficher toutes les branches du repository | git branch | |  |  |
| branch | créer une nouvelle branche | git branch maBranche | |  |  |
| checkout -b | créer une nouvelle branche et se positionner dessus en une seule instruction | git checkout -b maNouvelleBranche | |  |  |
| branch -d | supprimer la branche | git branch -d maBrancheASupprimer | |  |  |
|  |  |  | |  |  |
|  |  |  | |  |  |
| Fusionner les branches |  |  | |  |  |
| merge | fusionner la branche indiquée en paramètre sur la branche actuelle | git merge maBrancheAFusionner | |  |  |
|  | en cas de conflit, les résoudre manuellement dans les fichiers, puis : | git add monFichierConflictuel  git commit -a | | *Sinon, pour annuler la fusion :*  git merge --abort |  |
|  | ou utiliser une interface graphique pour résoudre le conflit avec | git mergetool monInterfaceGraphique | |  |  |
| rebase | déplace tous les commits de la branche actuelle sur le dernier commit de la branche indiquée en paramètre | git rebase master | |  |  |
|  | en cas de conflit, les résoudre manuellement un par un, puis : | git add monFichierConflictuel  git rebase --continue | | *Sinon, pour annuler la fusion :*  git rebase --abort |  |
|  |  |  | |  |  |
|  |  |  | |  |  |
|  |  |  | |  |  |
| Clé SSH |  |  | |  |  |
| ssh-keygen | génère une clé SSH à utiliser pour l'identification entre la machine et le compte GitHub | *Dans de dossier* Utilisateurs/Moi/.ssh  $ ssh-keygen -t rsa -b 4096 -C 'monadresse@sfr.fr' | |  | une fois générée, copier le contenu de id\_rsa.pub sur GitHub > settings > SSH  rien d'autre à faire sur la machine locale |
|  |  |  | |  |  |
| Remote repository | dépos à distance (via Github par exemple) |  | |  |  |
| clone | télécharge le repository distant complet sur la machine | git clone adresse/duRepository.git | |  |  |
| remote | affiche la liste des serveurs à distance | git remote | |  |  |
| remote add | sélectionne le repository distant via son adresse | git remote add origin git@github.com:nom/adresse.git | | git remote add monRemoteRepository git@github.com:nom/adresse.git | Rq: origin est le nom conventionnellement donné pour le repository remote principal. |
| remote show | récupère les données du serveur distant indiqué en paramètre | git remote show origin | |  |  |
| push -u origin | pousse le contenu validé (le/les commit(s) validés sur la machines mais absents sur le serveur) vers le repository distant  pour la branche en question  toutes les branches suivies (cad ayant été déclarées avec l'option -u) | git push -u origin maBranche  git push | |  | Rq: -u est un raccourcit de --set-upstream qui permet de mettre en place le suivit entre le repository local et distant pour la branche en question. |
| fetch | affiche les différences entre les commits sur le serveur et ceux sur la machine | git fetch origin | |  |  |
| git pull | télécharge sur la machine les commits récents de  la branche en question  toutes les branches suivies (cad ayant été déclarées avec l'option -u) | git pull origin master  git pull | |  |  |
| checkout -b pull-sample | télécharge une branche sur le serveur qui n'existe pas encore en local | git checkout -b pull-sample origin/maBrancheRemote | |  |  |
|  |  |  | |  |  |
|  |  |  | |  |  |
|  |  |  | |  |  |
|  |  |  | |  |  |
|  |  |  | |  |  |
|  |  |  | |  |  |
|  |  |  | |  |  |
|  |  |  | |  |  |
|  |  |  | |  |  |
|  |  |  | |  |  |
|  |  |  | |  |  |
|  |  |  | |  |  |
|  |  |  | |  |  |

PULL-REQUEST

Pour contribuer sur un projet open source :

1. Trouver le projet sur GitHub
2. Lire le fichier README pour connaitre les modalités de contributions à la section "Contributing" ou Pull-Request"
3. Fork le projet sur notre propre compte GitHub
4. Cloner le fork à partir de notre compte sur notre machine (git clone …)
5. (Il est possible de synchroniser la branche master de notre repository local avec, non pas le fork, mais la version officielle du repo pour prendre en compte toutes les récentes modifs

pour cela

* 1. on définit un nouveau remote : git remote add upstream https://github.com/...
  2. on se place sur la branche master
  3. on la synchronise : git branch --set-upstream-to=upstream/master )

1. Créer une nouvelle branche et travailler sur le code
2. Push la nouvelle branche sur notre fork
3. Créer une Pull-Request à partir de GitHub pour soumettre les modifications au projet officiel