GIT TUTOR

clone

Serveurs Git

- https://github.com (dépôt privé payant)
- https://bitbucket.org (dépôt privé gratuit)
- https://gitlab.org (open source)

Configuration

Aide

```
git config
```

Projet | *Utilisateur* | *Système*

```
git config [ -- [ local | global | system ] ] -l
```

Emplacements

- local : .git/config
- global : ~/.gitconfig
- system : /etc/gitconfig

Lecture | *Ecriture*

nécessaire à la première utilisation:

```
git config --get user.name
git config --get user.email

git config --add user.name inuitLarson
git config --add user.email francois@larsonneur.fr
```

Alias

```
git config --global alias.st status
```

```
git config --global alias.co checkout
```

permet le raccourci :

```
git st[atus]
```

directement dans ~/.gitconfig

```
[alias]
    st = status
    co = checkout
    l = log --pretty=oneline --abbrev-commit --graph --decorate --
    b = branch
    cg = config --global
    a = add
    c = commit
    cm = commit
    cgl = config --global -l
```

directement dans ~/.bash_profile

```
alias gst='clear && git st'
alias glo='clear && git log'
alias gco='git commit'
alias gb='git branch'
alias gbc='git checkout'
alias merge='git merge'
alias push='git push'
```

commit template message

```
git config --global commit.template=~/.gitmsg.txt
```

commit message editor

```
git config --global core.editor "subl -n -w"
```

Au préalable dans *.bashprofile*, ajouter :

```
export PATH="~/bin:$PATH
export EDITOR='subl -w'
```

Créer le lien symbolique : ln -s “/Applications/Sublime Text <n °version>.app/Contents/SharedSupport/bin/subl” ~/bin/subl

.gitignore dans un répertoire du dépôt

Exclure des fichiers du suivi :

```
# tout le répertoire app/uploads
app/uploads/*
```

Tout ignorer sauf certains fichiers :

```
# ignorer tout
*

# sauf ...
!.gitignore
!*.js
# etc...

# ...même dans les sous répertoires
!*/
```

global .gitignore (tous les dépôts)

```
git config --global core.excludesfile ~/.gitignore_global
```

Par exemple:

```
*~
*.orig
.DS_Store
log.txt
```

Repository

Nouveau

```
git init
```

Projet existant

```
git clone [url]
```

Historique

Par défaut HEAD :

```
git log [REF]
git show [ REF ]
```

où REF = HEAD | 40 caractères (chawan!)

Suivi d'un fichier :

```
git log --follow -p -- [pathfile]
gitk [pathfile]
```

Différence

De la copie locale avec la copie indexée :

```
git diff
```

De la copie locale avec le dernier commit :

```
git diff HEAD
```

De la copie indexée avec le dernier commit :

```
git diff --staged
```

Entre les sommets de 2 branches :

```
git diff b1 b2
```

Entre les sommets d'une branche et la branche de travail :

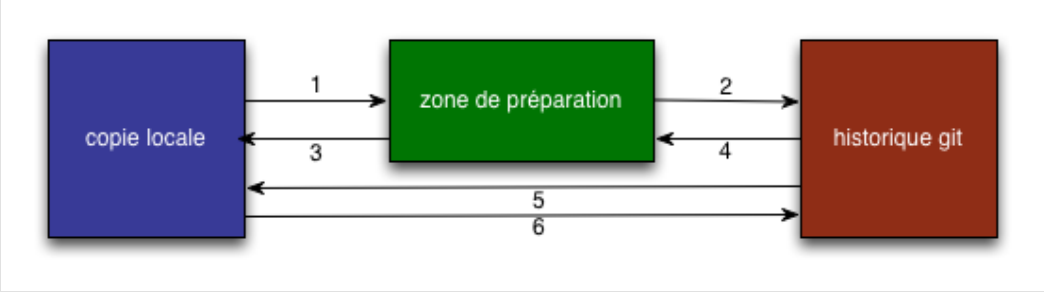
```
git diff b
```

Limiter la sortie aux noms de fichiers qui ont changés :

```
git diff --stat
```

***Changements d'états d'un fichier:**

```
non indexé (untracked) | modifié (modified [not staged]) | indexé
```



1 (ajouter | supprimer | déplacer un fichier dans la zone de préparation)

```
git add [file]
git add -u                ( fichiers connus par git )
git rm [--cached] file    ( suppression du git [pas du disque] )
git mv file               ( déplacement )
```

2 (commiter les modifications [d'un fichier])

```
git commit [file]
git commit -m "message" [file]
```

3 (ramener la zone de préparation dans la copie locale)

```
git checkout -- <file>
```

4 (ramener HEAD dans la zone de préparation | copie locale)

```
git reset --soft HEAD^
git reset --hard HEAD^
```

5 = 4 & 3 (ramener HEAD dans la copie locale)

```
git checkout HEAD <file>
```

6 = 1 & 2 (comité directement une modification)

```
git commit -a[m "message"] [file]
```

Remplacer la pointe courante de la branche par un nouveau commit

```
git commit --amend
```

⇔

```
git reset --soft HEAD^
... do something else to come up with the right tree ...
git commit -c ORIG_HEAD
```

Générer une nouvelle clé SSH

lister mes clés SSH

```
ls -al ~/.ssh
```

nouvelle clé SSH

```
ssh-keygen -t rsa -b 4096 -C "inuit.larson@orange.fr"
```

puis les réponses par défaut.

copier la clé public vers Github → settings → SSH keys)

```
cat ~/.ssh/id_rsa.pub
```

cloner en utilisant le protocole SSH (+sûr+rapide)

```
git clone "git@github.com:InuitLarson/Realtime-Markdown-Viewer.git
```

Exporter un repository (dans un répertoire existant)

```
git archive master | tar -x -C ~/export
git archive master | gzip > export.tar.gz
```

Tagger un commit

créer

tag léger (local)

```
git tag TAG [REF]
```

tag lourd (lourd)

```
git tag -a [-m MESSAGE] TAG [REF]
```

supprimer

```
git tag -d TAG
git push --delete REMOTE TAG
```

publier

```
git push --tag
git push REMOTE TAG
```

Gestion des branches

lister

Par défaut, les branches locales :

```
git branch [ -r(emote) | -a(ll) ]
```

information de tracking

```
git branch -vv
```

créer

En local, depuis une branche distante (crée la branche de même nom si le nom local [bch_loc] est omis) :

```
git checkout --track remote/branch [bch_loc]
```

À distance, depuis une branche locale (crée la branche de même nom si le nom distant [bch_dist] est omis):

```
git push -u remote[/bch_dist] branch    // -u ↔ --set-upstream
```

Sans changer de branche :

```
git branch BRANCH [REF]
```

En changeant de branche :

```
git checkout -b BRANCH [REF]
```

équivalent à :

```
git branch BRANCH [REF]
git checkout BRANCH
```

Attention : Modification du dépôt local

supprimer

La branche doit être fusionner :

```
git branch -d BRANCH
```

sinon :

```
git branch -D BRANCH
```

Attention : Les commits de la branche sont orphelins. Ils seront supprimer sous 90 jours (valeur par défaut du délai d'optimisation du dépôt Git).

récupérer des branches distance

```
git fetch branch
```

fusionner

```
git merge BRANCH
```

résoudre des conflits

```
git diff
```

```
git mergetool
```

Modifier l'outil de résolution de conflit :

```
git config --global merge.tool kdiff3
```

Sous Mac, avec l'***alias.cg = "config –global"*** :

```
git cg mergetool.kdiff3.path "/Applications/kdiff3.app/Contents/Ma
```

annuler la résolution d'un conflit

Revenir avant le merge:

```
git reset --hard HEAD^
```

Synchronisation

Publier

```
git remote
git push -u  remote branch      // -u ↔ --set-upstream
```

Publier la branche *master* sur le remote *origin* :

```
git push -u origin master
```

Publier la branche en cours sur *origin* :

```
git push
```

Récupérer les informations des branches distance

Informations de tracking :

```
git branch -vv
```

Toutes les branches d'une remote

```
git fetch remote
```

Toutes les branches de toutes les remote

```
git fetch --all
```

Récupérer une branche distante d'une *remote*

```
git pull remote branch  // pull = fetch + merge
git pull    // avec les informations de tracking
```

Hack Fork Github

Github ne permet pas de "forker" (fourchetter!) simplement un de ses dépôts.

```
git clone git@github.com:InuitLarson/depot.git depot-fork
cd chemin/depot-fork
vim .git/config
[remote "origin"]
        fetch = +refs/heads/*:refs/remotes/origin/*
        url = github.com:InuitLarson/depot.git depot
        #remplacer depot par depot-fork
git remote add upstream github.com:InuitLarson/depot.git depot
git push -u origin master
git remote
```

Premier commit pour Bitbucket

```
cd ~/GIT/depot
git init
git remote add origin git@bitbucket.org:uid_bitbucket/depot.git
touch 'FL'  > contributeurs.txt
git commit -am "Premier commit avec contributeurs.txt"
git push -u origin master
```

Hook : prepare-commit-msg

```
cd depot/.git/hooks/
mv prepare-commit-msg.sample prepare-commit.msg
```

Copier dans le presse-papier :

```
#!/bin/bash
set | egrep GIT > /dev/null
#echo AUTEUR is $GIT_AUTHOR_NAME

branchPath=$(git symbolic-ref -q HEAD)  #Somthing like refs/heads/
branchName=${branchPath##*/}            #Get text behind the last

firstLine=$(head -n1 $1)

# Check that this is not an amend by checking that the first line
if [ -z "$firstLine" ] ;then
    # $1 === $(git st)
    # Suppression du # de commentaire si tabulation
    sed -i '.bak' 's/#[TABUL]//' $1
    # Ajout de l'auteur et de la branche en cours
    sed -i '.bak' "1s/^/[$GIT_AUTHOR_NAME → $branchName]/" $1
fi
```

Dans un ancien dépôt utilisant déjà le hook :

```
pbcopy < ~/GIT/old_depot/.git/hooks/prepare-commit.msg
```

Remplacer le contenu de **prepare-commit.msg** :

```
pbpaste > prepare-commit.msg
```

Attention :

1. **pbcopy** et **pbpaste** sur *Mac OSX* seulement.
2. Remplacer [TABUL] par le caractère **tabulation**