Table des matières

2019-10-17: Ajout commit guidelines
2020-08-15: Sortie console sans *less*

Configuration de git:

La Doc Git : Paramétrage à la première utilisation de Git

```
git config --global user.name "YOUR NAME"
git config --global user.email "YOUR EMAIL ADDRESS"
```

Pour voir la configuration :

```
$ git config --list
user.name=Scott Chacon
user.email=schacon@gmail.com
color.status=auto
color.branch=auto
color.interactive=auto
color.diff=auto
...
```

Pour ne pas avoir la sortie dans ***less***

```
git config --global pager.branch false
```

Quelques commandes:

Status & Log

```
git status
git log

# visu des logs sur une ligne:
$ git log --oneline

# voir les différences:
$ git diff code-ancien code-récent
```

Add: Ajout de fichier en surveillance Git

```
# ajout de tous les fichiers/dossiers
git add -A
```

```
git add .
```

Commit: Faire un point d'archive

```
# commit avec message
git commit -m "commentaire de modif/ajout"

# commit avec fichier extrait de changelog
# dans répertoire précédant pour ne pas être dans git
git commit -F ../commit.txt

# modif commentaire du dernier commit (merci @Sam)
git commit --amend
```

Extrait de la doc pour ***git commit -am "message"*** :

```
$ git help commit
[...]
OPTIONS
       -a, --all
           Tell the command to automatically stage files that have been
modified and deleted, but new
           files you have not told Git about are not affected.
```

Checkout & Branch: Continuer un Dev sans casser ce qui marche déjà !

La Doc Git en FR

Les-Les-branches-avec-Git-Branches-et-fusions

Créer une branche et y basculer

```
# Pour créer une branche et y basculer tout de suite,  git checkout avec
l'option -b :
$ git checkout -b prob53
Switched to a new branch "prob53"
# C'est un raccourci pour :
$ git branch prob53
$ git checkout prob53
```

Renommer une branche

```
# renomer une branche
git branch -m nom_actuel nouveau_nom
```

Supprimer une branche

```
# supprimer une branche
git branch -d prob53
```

Branches locales/remote

```
# voir les branches locales
$ git branch -v
  doc    1afaa9e Ajout doc.txt avec la description de l appli
* master 307df92 Merge de test vers master
  test   6baf854 archive.py: Suppression des methodes Play/Pause/Stop

# voir les branches locales + remote
$ git branch -a
  doc
* master
  test
  remotes/open1024/doc
  remotes/open1024/master
  remotes/open1024/test


# Pour merger 2 branches: "Greffe de la branche test dans master"
# 1: Se placer dans la branche cible, ex: master
$ git checkout master
# 2: Commande de merge:
# (--no-ff no fast forward pour obliger à créer un point de commit même si peu de modifs)
$ git merge --no-ff -m "message de commit" test
```

Branches suite à clonage

```
# Gestion des branches suite à clonage:
git checkout -b doc origin/doc
git checkout -b test origin/test
```

.gitignore

Ce fichier permet de ne pas suivre des fichiers et dossiers.

```
.idea/
venv/
```

Exemple de **.gitignore** en fonction des langages utilisés :
github.com/github/gitignore

Pour **Python** voir Fichier .gitgnore pour Python

Coloration Git

```
# coloration du terminal Ex: git status (merci @Thierry)
git config color.ui true
```

Les alias

```
git config --global alias.co checkout
git config --global alias.br branch
git config --global alias.cm commit
git config --global alias.st status
git config --global alias.last 'log -1 HEAD'
git config --global alias.l 'log --oneline --graph -10'

# pour écraser un alias: --replace-all
$ git config --global alias.l 'log --oneline' --replace-all
```

Repositories distant

Remote: Ajout d'un repo distant

```
# ajout d'un dépôt
git remote add origin https://github.com/JC-onLine/Serial_SQL_Logger.git
git remote show origin
git remote remove origin
```

Init --bare: Init d'un Repository distant

Coté serveur :

Extrait de la **Doc Git en FR :** Git-sur-le-serveur-Mise-en-place-du-serveur
Une fois le ssh étabie.
[...]
Maintenant, vous pouvez créer un dépôt vide nu en lançant la commande ***git init*** avec l'option ***–bare***, ce qui initialise un dépôt sans répertoire de travail :

```
$ cd /opt/git
$ mkdir projet.git
$ cd projet.git
$ git --bare init
```

Coté local :

```
# Sur l'ordinateur de John
$ cd monproject
$ git init
$ git add .
$ git commit -m 'premiere validation'
$ git remote add origin git@gitserveur:/opt/git/projet.git
$ git push origin master
```

Les autres Giteurs

À présent, les autres utilisateurs peuvent cloner le dépôt et y pousser leurs modifications aussi simplement :

```
$ git clone git@gitserveur:/opt/git/projet.git
$ cd projet
$ vim LISEZMOI
$ git commit -am 'correction fichier LISEZMOI'
$ git push origin master
```

Push: Transfert vers le repo distant

```
# transfert vers dépôt
git push origin master
```

```
# transfert vers dépôt avec tracker "-u"
git push -u origin master
```

Pull: Reprise depuis repo distant

```
# reprise depuis dépôt
git pull origin master
```

```
# pour reprendre depuis dépôt et écraser le git local (merci @Sam)
git pull --rebase
```

Jargon Git Hub

- **Issues**: Ce sont des problèmes.
- **Milestones**: Ce sont des objectifs.

Tag: Version de l'appli

Créer un tag

```
# Créer l'indice de version
$ git tag v0.1.0

# Créer l'indice de version avec annotation
$ git tag -a v0.1.0 -m "Première version from scratch"
```

voir un tag

```
$ git show v0.1.0
```

Voir la liste des Tag

```
# Voir la liste des versions
$ git tag
v0.1.0
```

Transfert d'un Tag Release vers repo

```
# Transférer cette versions (tag v0.1.0) vers de repo distant
$ git add -A
$ git commit -m "Ready for seding release v0.1.0"
$ git push origine v0.1.0
To https://github.com/gCKn/test-gckn.git
* [new tag]         v0.1.0 -> v0.1.0
```

Supprimer un Tag Release

```
$ git push origin -d v0.1.0
To https://github.com/gCKn/test-gckn.git
- [deleted]         v0.1.0
```

Reset

Voir la doc en FR :
Utilitaires Git - Reset démystifié

Commit Guidelines

Voici des exemples de ligne de conduite d'Angular : CONTRIBUTING.md

```
docs(changelog): update changelog to beta.5
```

```
fix(release): need to depend on latest rxjs and zone.js

The version in our package.json gets copied to the one we publish, and users
need the latest of these.
```

Le protocole Angular est le suivant :

```
<type>(<scope>): <subject>
<BLANK LINE>
<body>
<BLANK LINE>
<footer>
```

Type

Must be one of the following:

- **build**: Changes that affect the build system or external dependencies (example scopes: gulp, broccoli, npm)
- **ci**: Changes to our CI configuration files and scripts (example scopes: Circle, BrowserStack, SauceLabs)
- **docs**: Documentation only changes
- **feat**: A new feature
- **fix**: A bug fix
- **perf**: A code change that improves performance
- **refactor**: A code change that neither fixes a bug nor adds a feature
- **style**: Changes that do not affect the meaning of the code (white-space, formatting, missing semi-colons, etc)

- **test**: Adding missing tests or correcting existing tests

Scope

The scope should be the name of the npm package affected (as perceived by the person reading the changelog generated from commit messages).

The following is the list of supported scopes:

- **animations**
- **common**
- **compiler**
- **compiler-cli**
- **core**
- **elements**
- **forms**
- **http**
- **language-service**
- **platform-browser**
- **platform-browser-dynamic**
- **platform-server**
- **platform-webworker**
- **platform-webworker-dynamic**
- **router**
- **service-worker**
- **upgrade**
- **zone.js**