Introduction to Git

Qu’est-ce git ? (french version)

Git est un logiciel de gestion de version decentralise en anglais Version Control System (VCS). C’est un logiciel qui aides les développeurs à travailler ensemble et a suivre l’historique de leur travail. Git est un logiciel libre créé par Linus Torvalds, auteur du noyau Linux.

Difference entre Git et SVN + TFS

SVN ou Subversion est un système de gestion de source centralise. C’est a dire qu’il faut un serveur pour des créer des branches et archiver du code ou tout se passe sur le serveur.

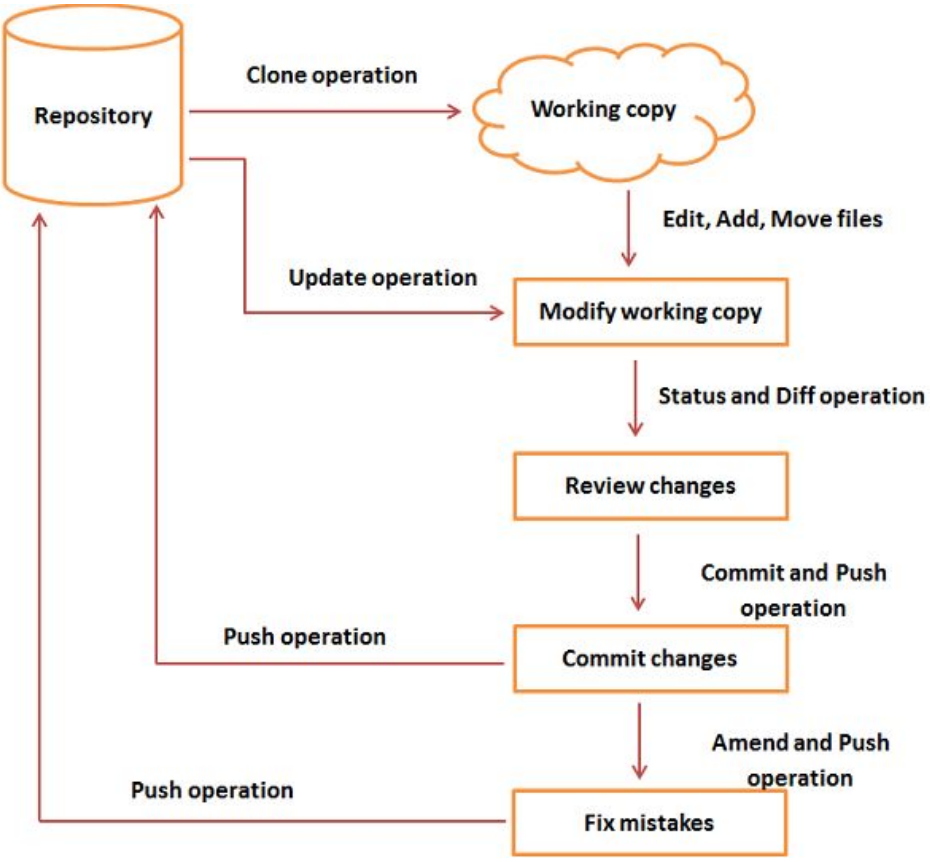
Git a l'opposé est un système de gestion de source décentralisée. On peut travailler directement en local sans serveur.

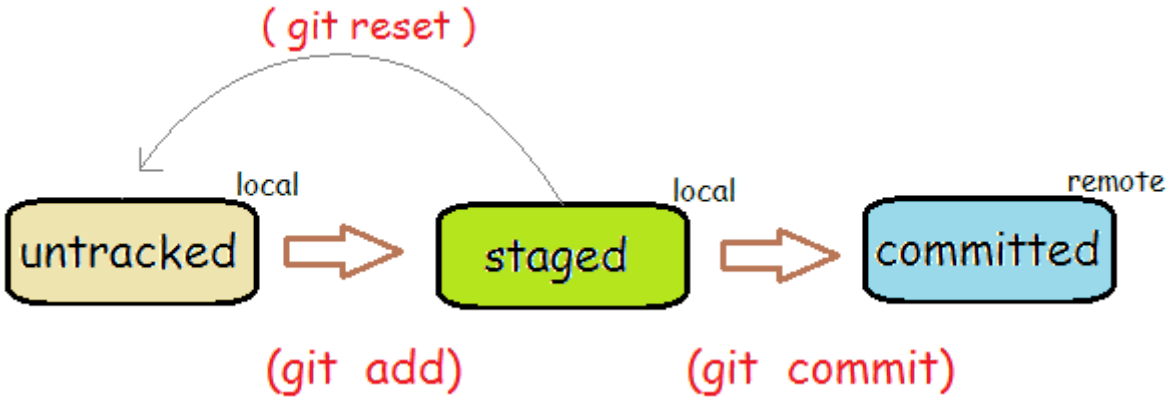
Pour comparer avec TFS, SVN et Git sont des logiciels de sources de controls seulement. Ce ne sont pas des logiciels pour faire des suivies de bugs, des rapports, de l'intégration continue, etc. TFS est un logiciel de source centralisée

Installation de Git

https://www.atlassian.com/git/tutorials/install-git

Cycle de vie


Repository
Clone operation
Working copy
Edit, Add, Move files
Update operation
Modify working copy
Status and Diff operation
Review changes
Commit and Push operation
Push operation
Commit changes
Amend and Push operation
Push operation
Fix mistakes


(git reset)
local
local
remote
untracked
staged
committed
(git add)
(git commit)

Git HTTPS et SSH

2 techniques pour copier le répertoire sur Github:

- HTTPS URLs

https:// clone URLs disponible sur tous les répertoires publique et prive. Quand tu fais un Git clone par exemple sur le répertoire distant, Git va te demander tes identifiants Github (nom d'utilisateur et mot de passe).

- SSH fournit un accès au répertoire distant via SSH, un protocole de sécurité. Tu dois générer une paire de clé privée entre ton ordinateur et une clé publique de ton compte Github. Quand tu fais un Git, si n'a pas entrer de "passphrase", tu n'auras pas besoin de saisir les identifiants.

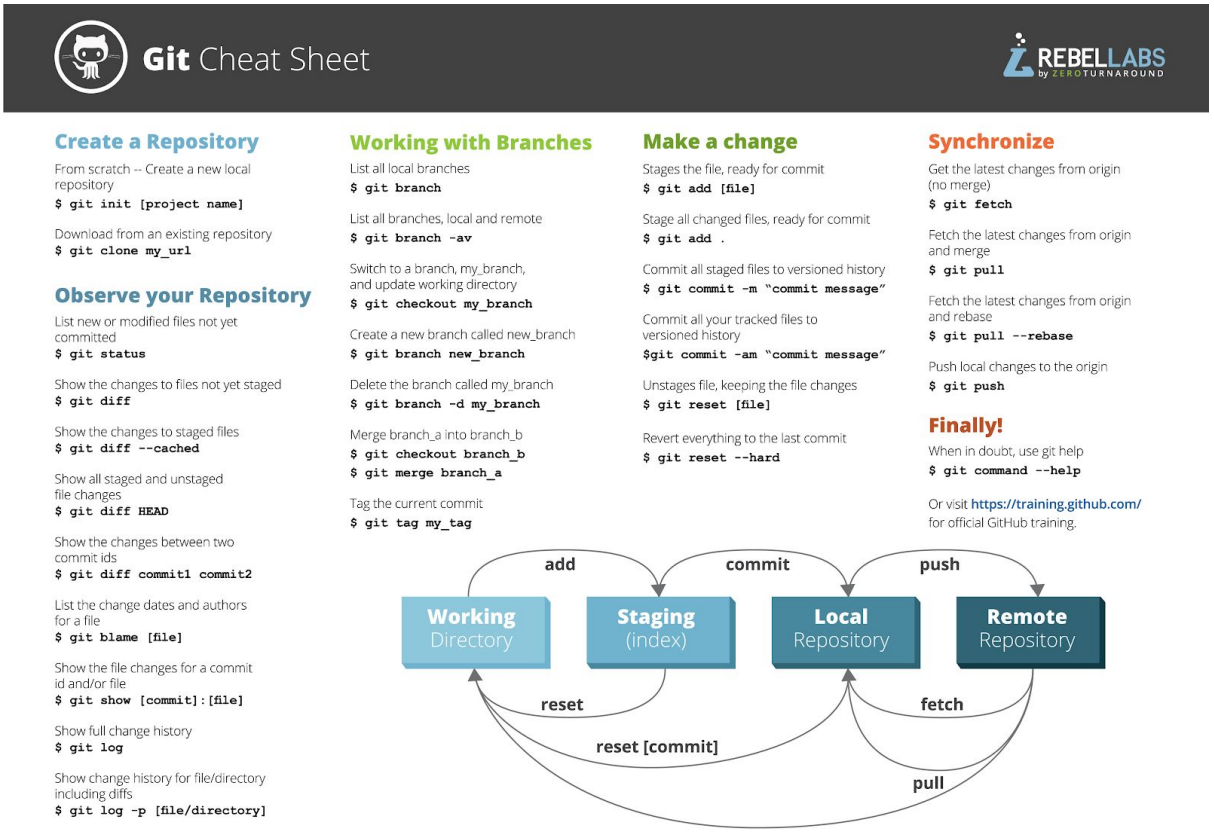
.gitignore

Le fichier .gitignore est généralement placé à la racine du projet et contient un pattern des fichiers à exclure du projet.

Documentation officiel

https://git-scm.com/docs

Cheat Sheet



https://i.redd.it/8341g68g1v7y.png
https://www.atlassian.com/git/tutorials/atlassian-git-cheatsheet
https://www.git-tower.com/blog/media/pages/posts/git-cheat-sheet/-1223884809-1590818205/git-cheat-sheet-large01.png
https://intellipaat.com/mediaFiles/2019/03/Git-Cheat-Sheet.jpg

Liens

https://github.com/
https://bitbucket.org/product
https://www.atlassian.com/git/tutorials

Pratique

Step 1 - init

```
echo "# tuto-git" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin git@github.com:pjserol/tuto-git.git
git push -u origin master

Note:
   -  README, information about project.
```

Step 2 - add, commit, pull & push

```
git add .
git add filename
git commit -m "second commit"
git pull
git push

Note:
   -  . to add all the files
   -  git pull to check if something changed on the master branch
```

User 2

```
User 1, settings -> manage access -> invite a collaborator
User 2:
   -   receive the invite, then accept the invite
   -  Create a folder struct for the project example
      (Users/pjserol/code/github.com/pjserol/tuto-git/)
```

Step 3 (user 2) - clone, status & config

```
git clone https://github.com/pjserol/tuto-git.git
// cd → command directory
cd tuto-git/
// list
ls
git status
// change a file
git add filename
// config user
git config --global user.email "you@example.com"
git config --global user.name "Your name"
```

```
git commit -m "first commit user 2"
git pull
git push
```

Step 4 - branch & checkout

```
git pull
git branch
git checkout -b Feature/01
git add filename
git commit -m "third commit"
git push --set-upstream origin Feature/01
```

Or

```
git pull
git branch
git branch Feature/01
git checkout Feature/01
git add filename
git commit -m "third commit"
git push --set-upstream origin Feature/01
```

Note: You can also go use Visual Studio Code to see what changes in the file

Step 5 - stash & pop

```
git checkout master
// change a file
// record the current state of the directory
git stash
git checkout Feature/01
// do some work (commit, push)
git checkout master
// remove a single stash from the list
git stash pop
git commit -m "third commit"
git push --set-upstream origin Feature/01
```

Step 6 - diff & log

```
// show the changes of the files
git diff
// show the files changes for a commit
git show
// show full change history
git log
// list of the changes dates and authors for a file
git blame index.html
```

Note: show with github, how to create a Pull Request

Step 7 - git reset

```
// change a local file
git add .
// unstage file
git reset
git add .
git commit -m "other commit"
// cancel local commit
git reset HEAD~1
```

Note: HEAD is the pointer to the current branch reference

Step 8 - git merge (and fix conflict)

```
git checkout master
// change a local file
git add .
git commit -m "other commit"
git push
git checkout Feature/01
// apply the last change of master in the branch
git merge master
```

Note: fix a conflict (:wq), fix the files with a conflict then you can commit and push your code

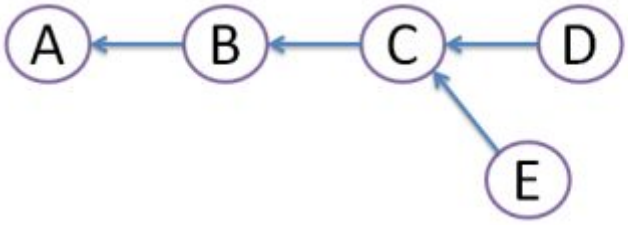
Step 9 - revert

```
git checkout master
// change a local file
git add .
git commit -m "other commit"
git push
git log
git revert 59ba0e7
git push
```

Difference between git merge and rebase
https://stackoverflow.com/questions/16666089/whats-the-difference-between-git-merge-and-git-rebase/25267150

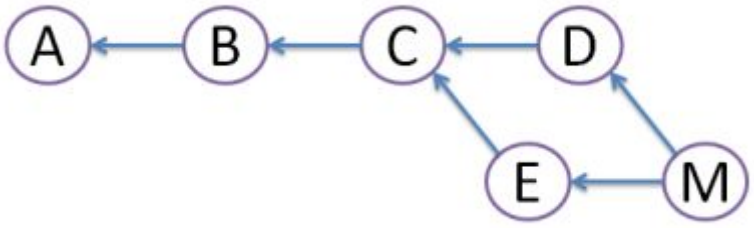
Suppose originally there were 3 commits, `A`,`B`,`C`:

Then developer Dan created commit `D`, and developer Ed created commit `E`:



Obviously, this conflict should be resolved somehow. For this, there are 2 ways:

MERGE:



Both commits `D` and `E` are still here, but we create merge commit `M` that inherits changes from both `D` and `E`. However, this creates *diamond* shape, which many people find very confusing.

REBASE: