

# Présentation de Git / GitHub

Joris Tillet

UV 5.8 – Ingénierie Système et Modélisation Robotique

18 décembre 2019

# Sommaire

---

- 1 Présentation & installation
- 2 Commandes de base
- 3 Dépôt distant
- 4 Branches
- 5 Contribuer dans un projet
- 6 Conclusion

## Intérêt de Git

But : suivre la vie d'un ou plusieurs fichiers.

- Quand le fichier a été modifié ?
- Quels sont les changements ?
- Pourquoi ces changements ?
- Qui en est à l'origine ?

⇒ Permet d'avoir plusieurs versions d'un même fichier, mais bien organisées.

# Différence entre Git et GitHub

---

## Définition (Wikipédia)

Git : Git est un logiciel (libre) de gestion de versions décentralisé.

## Définition (Wikipédia)

GitHub : GitHub est un site d'hébergement et de gestion de développement compatible avec Git.

- Git est un "VCS" (Version Control System).
- GitHub est le site de dépôt le plus répandu, mais il en existe d'autre (FramaGit, etc).

# Installation de Git

---

## Sur Linux

Git est disponible sur les dépôts officiels :

```
# apt install git
```

Ou sur <https://git-scm.com/downloads>.

## Sur Windows

Un installeur est disponible sur le site <https://msysgit.github.io/>.

- Certains éditeurs de texte ou IDE intègrent directement les VCS.

# Premières commandes

---

## Création d'un répertoire Git

- Soit à partir d'un dossier déjà existant (en local) :  
`$ git init`
- Soit à partir d'un projet existant déjà :  
`$ git clone <adresse_du_projet>`

## Status

```
$ git status
```

Indique l'état actuel du répertoire :

- Les nouveaux fichiers pas encore suivis,
- les changements non enregistrés,
- d'autres informations utiles sur ce que vous êtes en train de faire.

## Ajout d'un fichier

```
$ git add <chemin>
```

ou

```
$ git add -A
```

- Ajoute les fichiers de *chemin* à l'index de Git.
- L'option '-A' (ou --all) permet d'ajouter tous les fichiers du répertoire courant.



## Commit

```
$ git commit -m "message du commit"
```

- Enregistre les changements avec une explication associée,
- L'option '-m' permet d'écrire le message obligatoire associé,
- L'option '-a' permet d'ajouter les modifications des fichiers déjà dans l'index avant le commit (évite le *git add*).

## Log : montre les derniers commits

```
$ git log # options utiles : --oneline --color --graph  
$ git whatchanged  
$ git show
```

## Utiliser un dépôt distant

---

### Création d'un remote

```
$ git remote add <nom_remote> <url>
```

Si un *git clone* a été utilisé au début pour initialiser le répertoire git, alors la remote existe déjà sous le nom *origin*.

## Push

```
$ git push <nom_remote> <nom_branche>
```

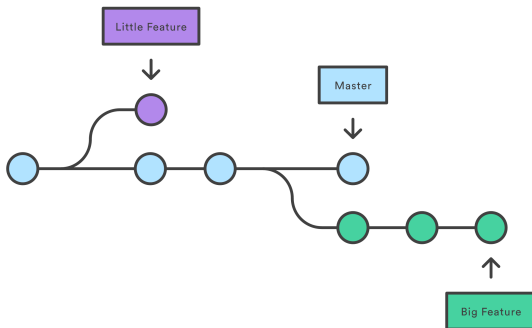
- Met à jour le dépôt distant.
- Il faut bien sûr avoir les droits sur ce dépôt, et cette commande demande souvent une identification (compte GitHub).

## Pull

```
$ git pull <nom_remote> <nom_branche>
```

- Met à jour le répertoire avec le dépôt distant associé à *nom\_remote*.
- Exécute en fait les deux commandes *git fetch* qui télécharge les données et *git merge* qui les fusionne avec le répertoire courant.
- Peut entraîner des conflits qu'il faut résoudre soi-même.

# Présentation des branches



- Permet de créer une nouvelle fonctionnalité sans casser le projet.
- La branche *master* est la branche de base. Elle devrait toujours contenir une version qui fonctionne.
- Peut servir à créer des "Tags", une version fonctionnelle qu'on souhaite garder.

# Utilisation des branches

## Checkout

```
$ git checkout <nom_branche>
```

- Permet de passer d'une branche à l'autre.
- L'option '-b' permet de créer une nouvelle branche (et d'y aller dessus),
- Utiliser *git branch --help* pour gérer les branches.

## Merge

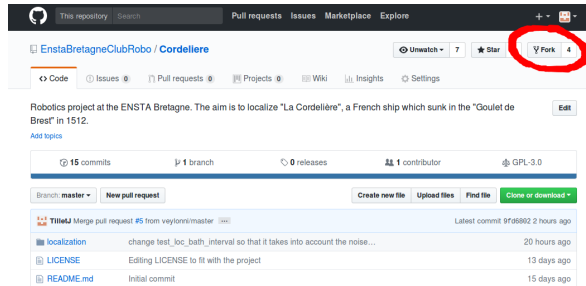
```
$ git merge <nom_branche>
```

Permet de fusionner la branche *nom\_branche* avec la branche actuelle.

# Contribuer dans un projet OpenSource

## Notion de fork

Sur GitHub, il est possible de créer un "fork" d'un projet existant.



- Permet de créer une copie d'un projet et de travailler dessus de son côté avant de proposer de nouvelles fonctionnalités.

## Résumé des étapes :

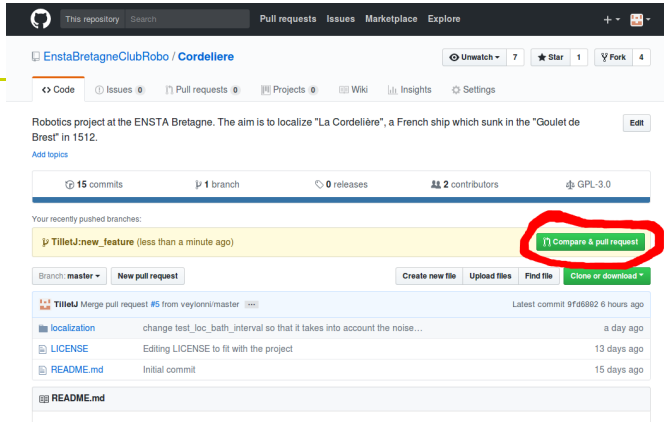
---

- Création d'un fork,
- Création d'un clone du fork en local,
- Création d'une nouvelle branche,
- Travail sur la nouvelle branche (commits),
- Mise à jour avec l'upstream,
- Push,
- Pull request.

### Pull Request

Une fois les nouvelles fonctionnalités développées, on peut les proposer au projet d'origine avec une "pull request".





EnstaBretagneClubRobo / Cordeliere

Robotics project at the ENSTA Bretagne. The aim is to localize "La Cordelière", a French ship which sunk in the "Goulet de Brest" in 1512.

15 commits 1 branch 0 releases 2 contributors GPL-3.0

Your recently pushed branches:

TilletJ:new\_feature (less than a minute ago) **Compare & pull request**

Branch: master New pull request Create new file Upload files Find file Clone or download

TilletJ Merge pull request #5 from veylonni/master Latest commit 9fd6892 6 hours ago

localization	change test_loc_bath_interval so that it takes into account the noise...	a day ago
LICENSE	Editing LICENSE to fit with the project	13 days ago
README.md	Initial commit	15 days ago

README.md

- Après le push, sur le GitHub du projet qu'on a forké, on a la nouvelle branche qui apparaît et GitHub propose d'en faire une pull request.
- Sinon il faut aller sur sa nouvelle branche dans GitHub et cliquer sur "Pull request".

## Mettre à jour son fork :

---

- Création d'un remote *upstream*,
- Fetch,
- Merge (ou rebase),
- Push.

### Fetch

Permet de télécharger les mises à jour du dépôt original dans une nouvelle branche *nom\_remote/nom\_branche*.

```
$ git fetch <nom_remote> <nom_branche>
```

```
$ git fetch upstream master # en général
```

## Mettre à jour son fork :

---

- Création d'un remote *upstream*,
- Fetch,
- Merge (ou rebase),
- Push.

### Merge & rebase

Permet de fusionner deux branches (intégrer les mises à jour).

```
$ git merge upstream/master
```

```
$ git rebase upstream/master # à préférer ici
```

Ajout des slides dans l'upstream du projet.

## Autres commandes *git*

---

- `$ git reset` # revient à l'index du dernier commit
- `$ git reset --hard` # impacte également les fichiers (attention)
- `$ git revert` # créer un commit qui annule un ancien commit
- `$ git reflog` # historique de tous les changements
- `$ git diff` # montre les changements entre commits, branches, etc
- `$ git rm / mv` # travaille sur le working tree et l'index
- `$ git stash` # met de côté le travail et revient au dernier commit
- `$ git stash pop` # récupère le travail mis de côté
- `$ git blame` # montre quelle ligne a été modifiée par qui, quand..
- `$ man git` # RTFM

# De l'utile en vrac

---

- Paramétrage de *git* :

- ▶ `$ git config --global user.name "<name>"`
- ▶ `$ git config --global user.email "<email@domain.bzh>"`
- ▶ `$ git config --global http.proxy <ip.address:port>`

- Outils utiles :

- ▶ IDE (Sublime, Atom, JetBrains suite, ...)
- ▶ ZSH (`# apt install zsh`) avec Oh My Zsh.
- ▶ Meld (`# apt install meld`)

- GitHub :

- ▶ Issues, Milestones
- ▶ @Mentions, #References
- ▶ Intégration (Taiga, Travis, ...)

- Très bonne gestion de versions,
- Compliqué au début,
- Meilleur ami ensuite.



FIGURE – Source : [xkcd.com](https://xkcd.com/152/)