# Présentation de Git / GitHub

Joris Tillet

ENSTA Bretagne

17 novembre 2017

# Sommaire

# Présentation

## Intérêt de Git

But : suivre la vie d'un fichier.

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

$\Rightarrow$ 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 dépot le plus répandu, mais il en existe d'autre (FramaGit, etc).

# Installation de Git

## Sur Linux

Git est disponible sur les dépots officiels :

```
$ sudo 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 IDLE intègrent directement les VCS.

# Premières commandes

## Création d'un répertoire Git

- Soit à partir d'un dossier déjà existant :

```
$ git init
```

- Soit à partir d'un projet existant déjà :

```
$ git clone adresse_du_projet
```

## Status

```
$ git status
```

Indique le statut actuel du répertoire :

- Les nouveaux fichiers pas encore suivis,
- et les changements non enregistrés.

## Ajout d'un fichier

```
$ git add nom_fichier
```

ou

```
$ git add -A
```

- Ajoute les fichiers 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.
- L'option '-m' permet d'écrire un message décrivant les changements (obligatoire),
- L'option '-a' permet d'ajouter les fichiers avant le commit (évite le *git add -A*).

# Utiliser un dépot 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

## Push

```
$ git push nom_remote
```

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

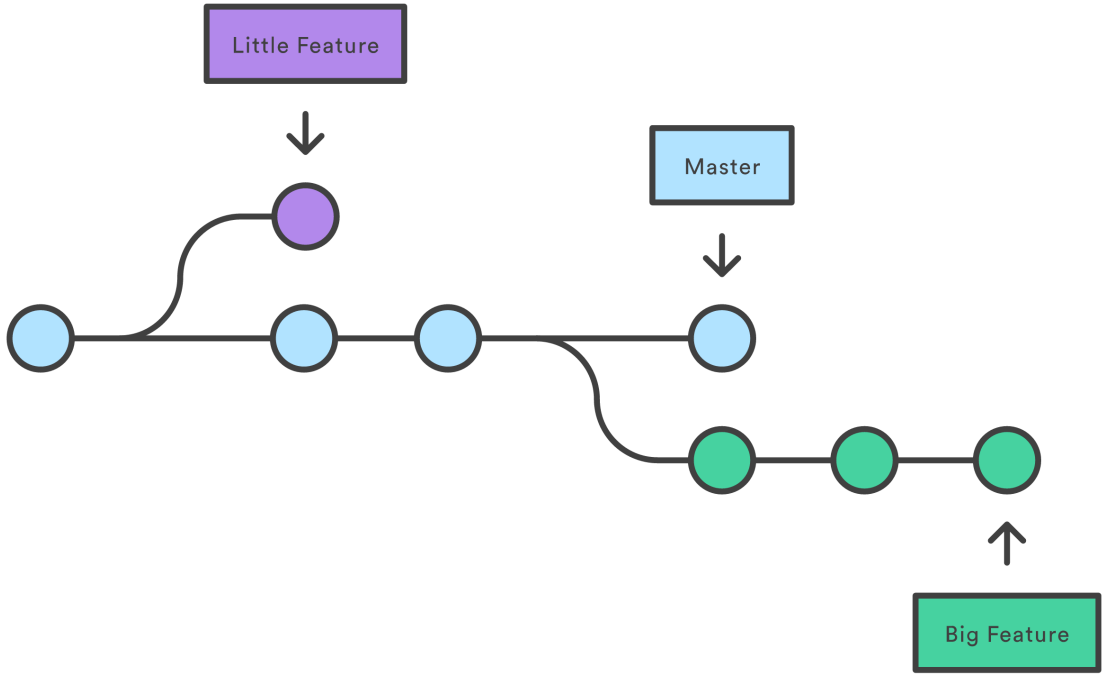
# Pull

## Pull

```
$ git pull nom_remote
```

- Met à jour le répertoire avec le dépot distant.
- Éxécute 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'aller dessus),
- Utiliser "*git branch − d nom_branche*" pour supprimer une branche.

## Merge

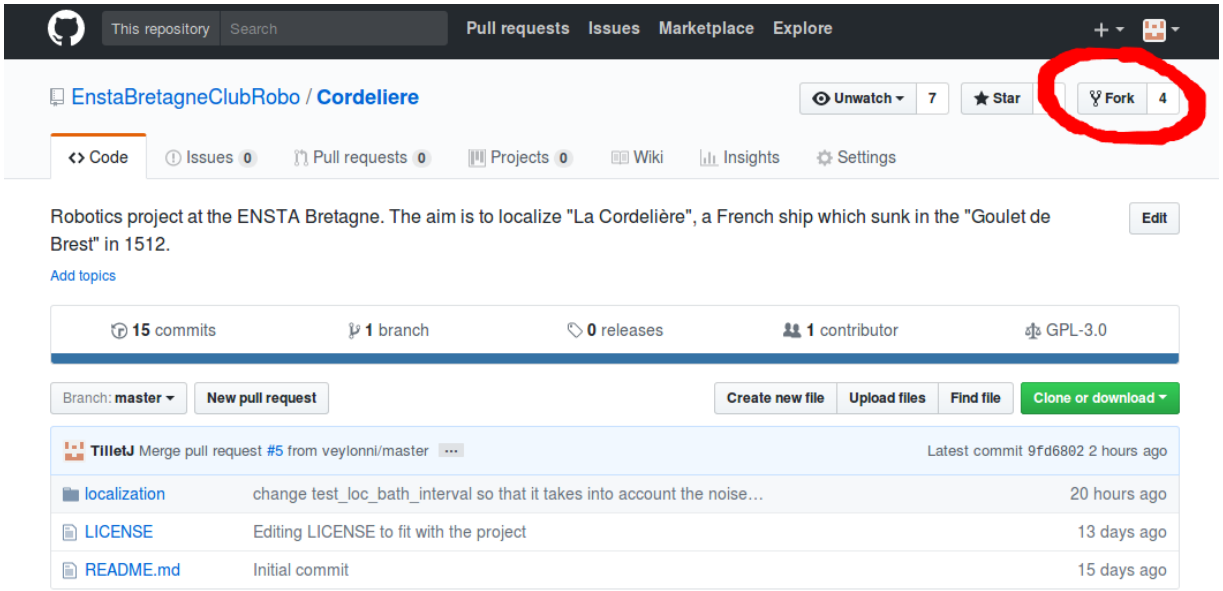
```
$ git merge nom_branche
```

- 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.




*Lien GitHub Cordelière*

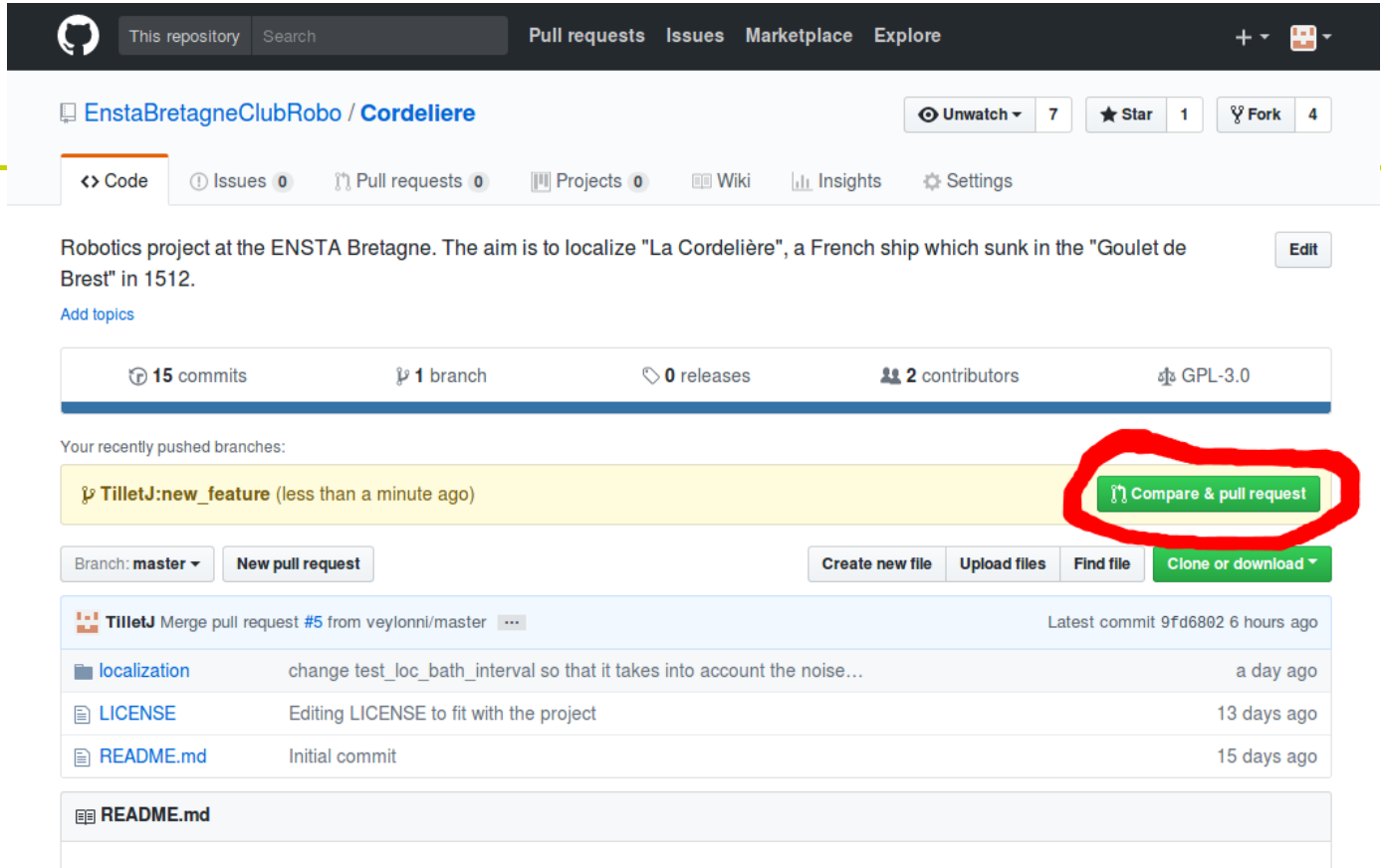
- Permet de créer une copie d’un projet et de travailler dessus de son côté avant de proposer de nouvelles fonctionnalités.

# 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,
- Commit & push,
- Pull request.

## Pull Request

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




- 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 un 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.

```
$ git fetch upstream
```

## Merge & rebase

Permet de fusionner deux branches.

```
          $ git merge upstream/master
ou        $ git rebase upstream/master # à préférer
```

# Exemple d'application

- Création d'un dossier *"Mini-Lessons"* avec la présentation dedans.

# Conclusion

- Très bonne gestion de versions,
- Compliqué au début,
- Indispensable pour un projet de grande ampleur.


FIGURE – Source : xkcd.com