RETOUR SUR LES FONDAMENTAUX

Retour sur les fondamentaux

Présentation générale

Git

- Logiciel Libre
- Crée par Linus Torvalds (2005) pour optimiser le noyau Linux
- Libre, pour remplacer bitKeeper
- Fonctionnement simple, rapide
- Pensé pour être décentralisé (Pair To Pair)
- Peut également être centralisé sur un serveur

- SVN ou CVS en mode centralisé = Une authentification est nécessaire pour travailler sur un seul dépôt

Git
Système de supervision et de versionnement d'un code source

GitHub

- ≥ + de 100 millions de repositories
- ≥ Premier commit en 2007
- ≥ Racheté par Microsoft en 2018 (stratégie open source)
- ≥ Permet de gérer des dépôts Git sur une interface web
- ≥ Interface et fonctionnalités enrichies en plus de Git :
  - ≥ Actions
  - ≥ Animations d'une communauté (réseau sociale)
  - ≥ Qualité de code
  - ≥ Statistiques, etc.

- ≥ D'autres solutions existent (ex: GitLab)

GitHub
Hébergement et gestion à distance (cloud) de dépôts Git

Retour sur les fondamentaux

Git Bash et outils GUI

Git Bash

≥ Permet d'utiliser Git en ligne de commande

≥ Installation

- Windows

https://gitforwindows.org/

- Type Linux (Debian / Ubuntu)

```
sudo apt install git
```

≥ Créer un projet (!!)

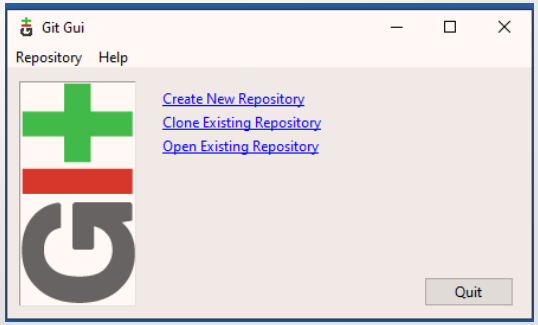
```
git init monSuperProjet
```

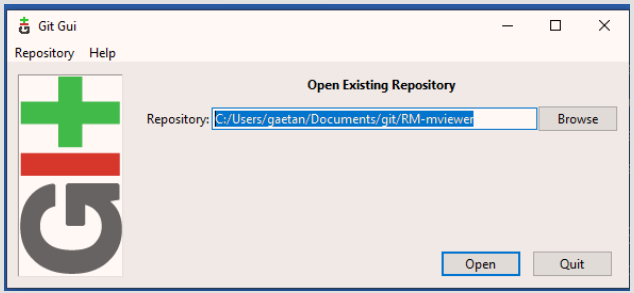
Bonus

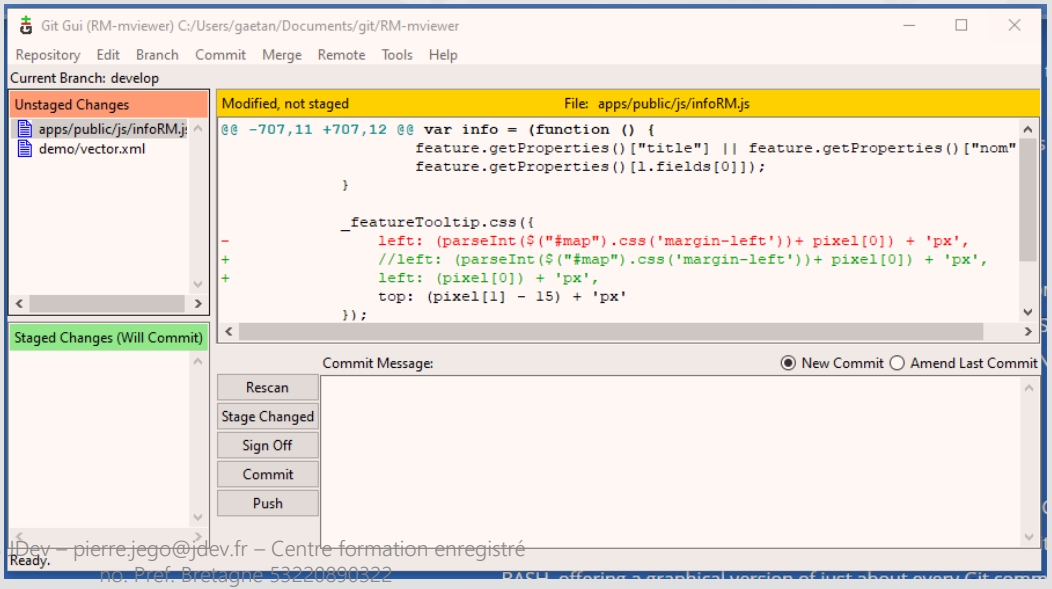
A l'installation, bien sélectionner « Git Bash Here » permet d'ouvrir un fichier dans Git Bash via un clic droit

Git GUI

- ≥ Propose une interface à la place des lignes de commandes
- ≥ Installé avec GIT





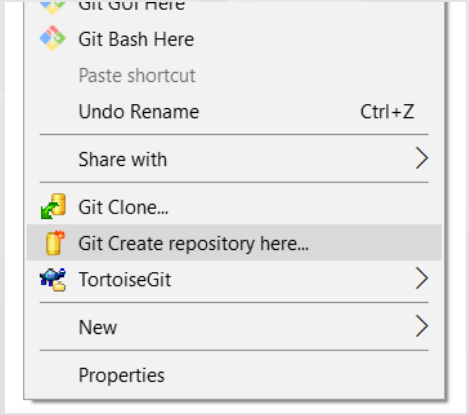


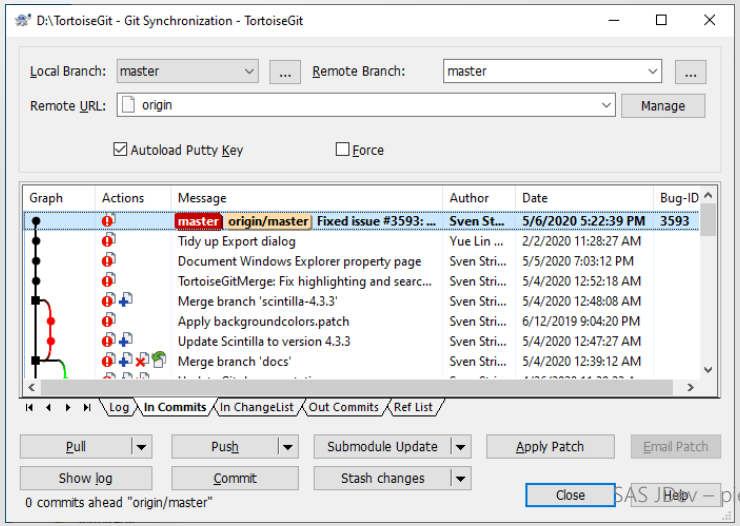
Turtoise git

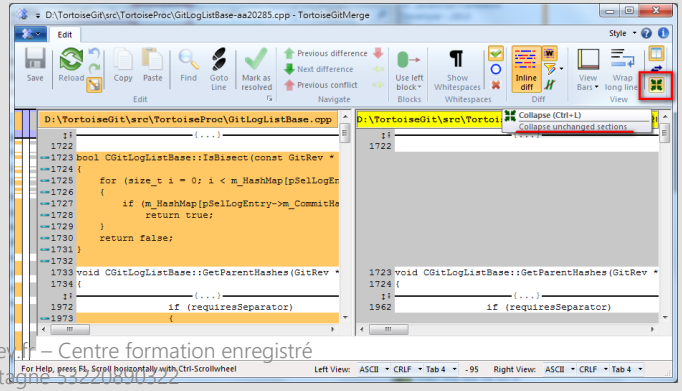
≥ Un accès raccourci via l'explorateur souris

≥ Une interface type Git GUI

https://tortoisegit.org/download/



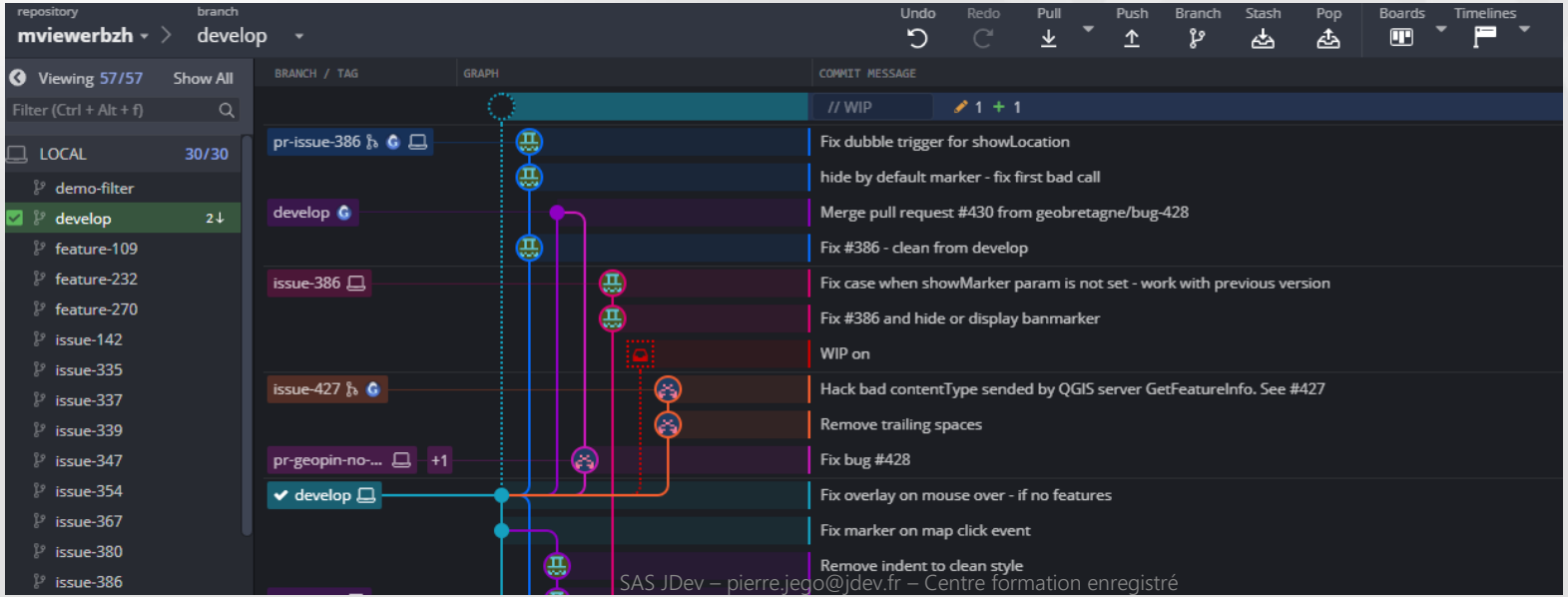




Git KRAKEN

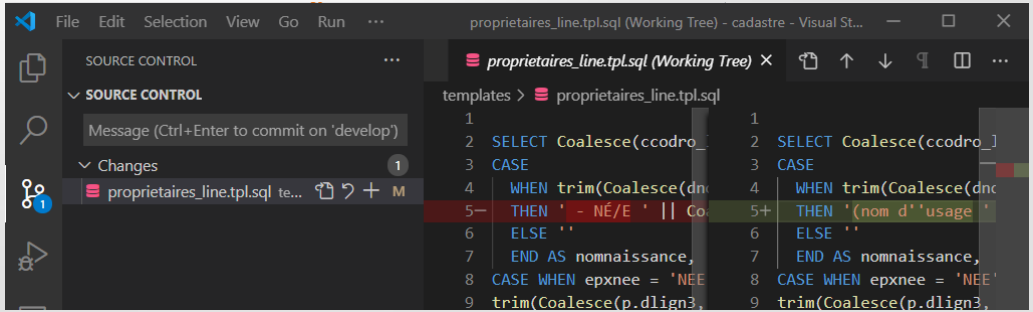
https://www.gitkraken.com/

- ≥ Propose une interface à la place des lignes de commandes
- ≥ GitHub en version Desktop (vue des branches, commits, fichiers, kanban, etc.)

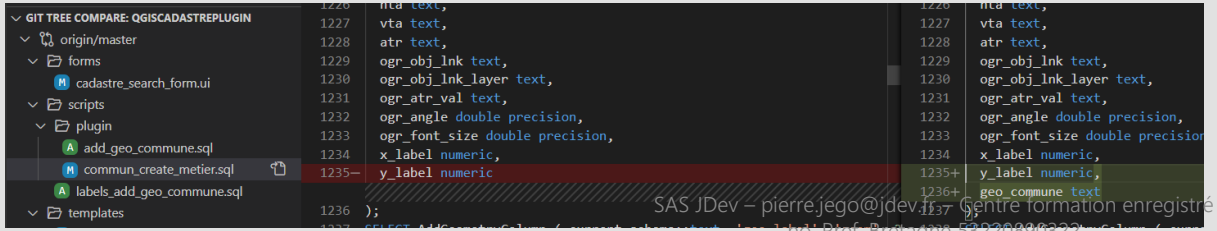


Plugins VSCode GIT

- ≥ VSCode contient des outils pratiques pour utiliser / superviser une ressource Git dans un environnement de développement
- ≥ Ils permettent de naviguer dans un dépôt sans ligne de commande
- ≥ Une interface type « terminal » intégré permet aussi d'exécuter des lignes de commande Git



- ≥ Le plugin Git Tree Compare permet de comparer les différences entre les branches d'un dépôt



Conseils

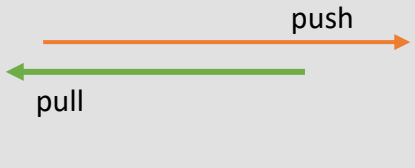
- Les outils bureautiques réalisent des commandes à votre place
- Il vaut mieux savoir ce que l'on réalise avec ses propres commandes (Git Bash)
- Il vaut mieux utiliser un outil avec lequel on se sent bien pour éviter des erreurs
- Les outils type Kraken ou Git Gui propose des modes de visualisation pratiques très utiles !

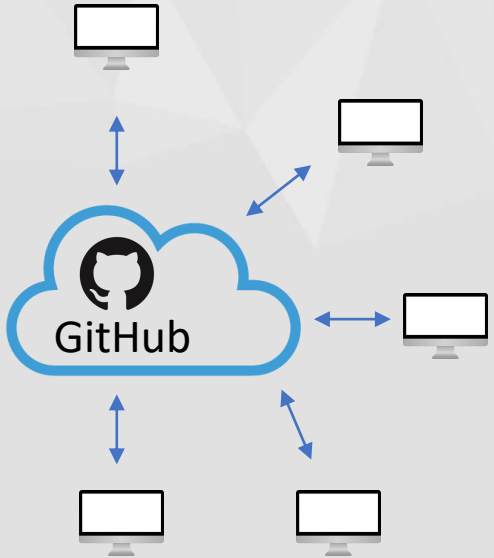
Retour sur les fondamentaux

Manipulations de base

Git & Github : Local & remote

- Il faut différencier le local du distant (remote)
- Le local représente le code à un instant T, il vit seul
- Le distant est « vivant » il est modifié par d'autres indépendamment de votre local
- Des actions sont nécessaires pour maintenir son état local ou partager des modifications



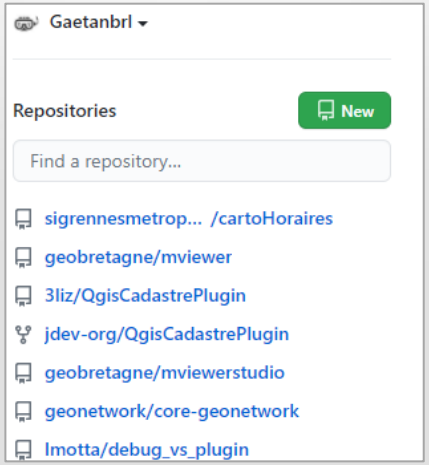


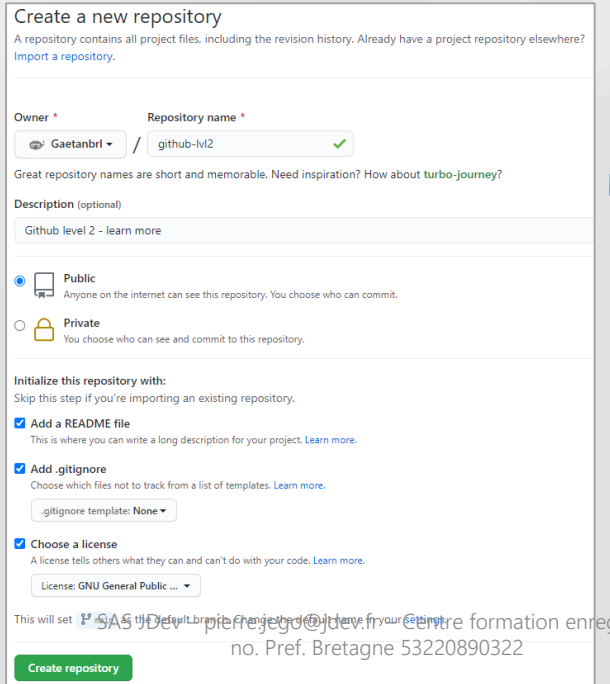
Votre PC
= Local

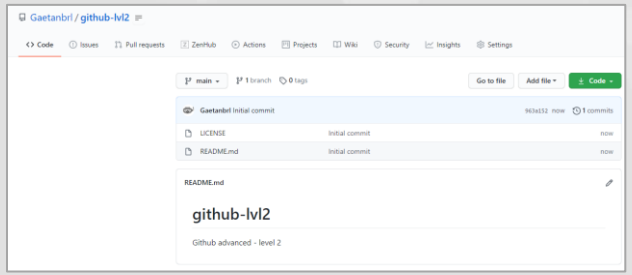
Distant

Github – Créer un projet

≥ Création d'un projet au sein d'une organisation ou d'un compte utilisateur (même procédure)



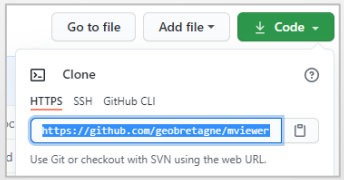




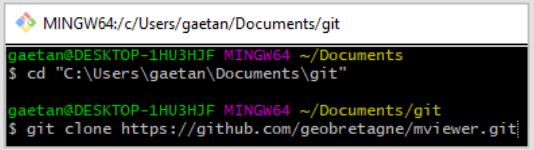
Github – Récupérer un projet en local

≥ Pour récupérer un code source sur votre ordinateur, vous devez utiliser git pour réaliser un « clone » local sur le disque dur

1. Récupérer l'URL du projet



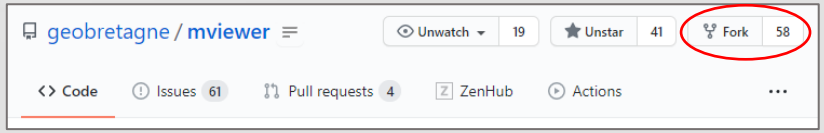
2. Se positionner dans un répertoire et exécuter la commande de « clone »

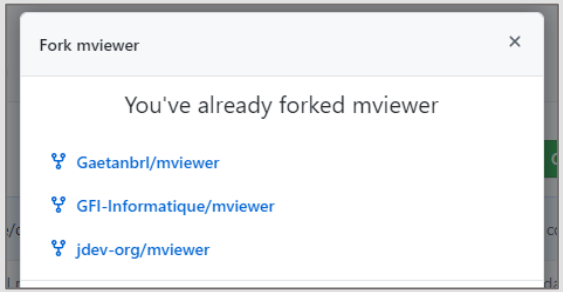


Github – Faire un fork

≥ Le fork permet de récupérer un projet vers un compte afin d'avoir une version « indépendante » et modifiable **sans conséquences**
≥ Il faut utiliser le bouton « Fork » d'un projet et sélectionner ensuite l'organisation / le compte ciblé

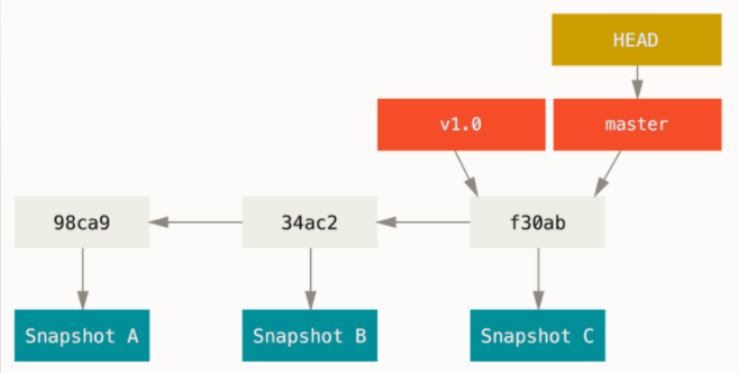
Un fork est un instance d'une version d'un code source.
Seule une action manuelle permet de garder un fork à jour.





Github – Créer une branche

- Une branche est une « copie » d'un code source d'un « espace » vers un autre « espace » parallèle
- Par défaut un projet est crée avec une seule branche souvent appelée « master » (le nom peut être changé à la création du projet)
- Les branches peuvent être consultées et crées via Git ou dans GitHub directement
- Une branche est toujours crée depuis une première branche sélectionnée



Github – Créer une branche

- ≥ Une branche est une « copie » d'un code source d'un « espace » vers un autre « espace » parallèle
- ≥ Par défaut un projet est crée avec une seule branche souvent appelée « master » (le nom peut être changé à la création du projet)
- ≥ Les branches peuvent être consultées et crées via Git ou dans GitHub directement
- ≥ Une branche est toujours crée depuis une première branche sélectionnée

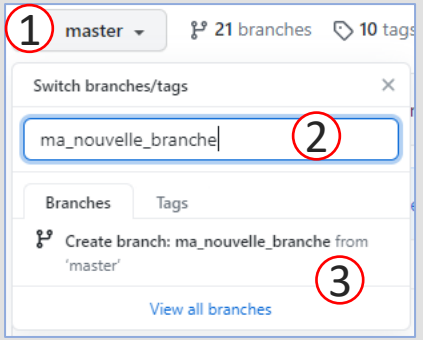
1. On sélectionne la branche de base

```
MINGW64:/c/xampp/htdocs/mviewerbzh
gaetan@DESKTOP-1HU3HJF MINGW64 /c/xampp/htdocs/mviewerbzh (develop)
$ git checkout -b ma_nouvelle_branche
```

2. On créer sa branche

```
MINGW64:/c/xampp/htdocs/mviewerbzh
gaetan@DESKTOP-1HU3HJF MINGW64 /c/xampp/htdocs/mviewerbzh (develop)
$ git branch -a
  demo-filter
* develop
```

Avec GitHub



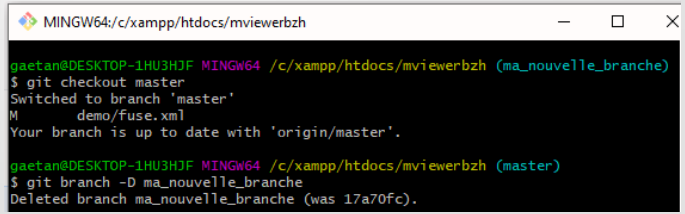
3. Avec Git on doit pousser la branche sur le serveur distant (remote)

```
MINGW64:/c/xampp/htdocs/mviewerbzh
gaetan@DESKTOP-1HU3HJF MINGW64 /c/xampp/htdocs/mviewerbzh (develop)
$ git push origin ma_nouvelle_branche
```

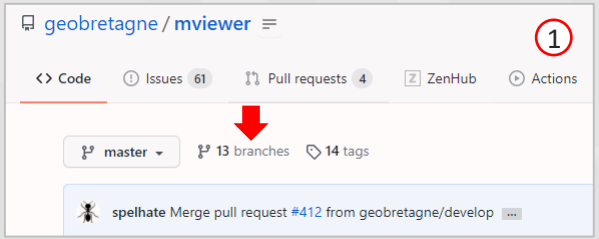
Github – Supprimer une branche

≥ Une branche peut être supprimée avec Git ou directement dans GitHub
≥ Dans TOUS les cas, une action est nécessaire dans Git

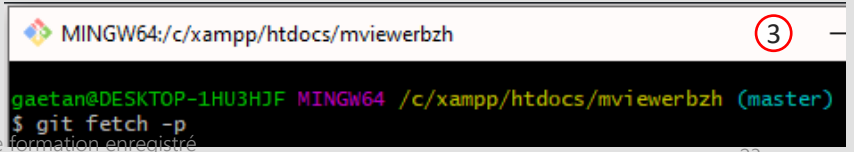
1. Se positionner sur la branche master
2. Supprimer la branche



Avec GitHub

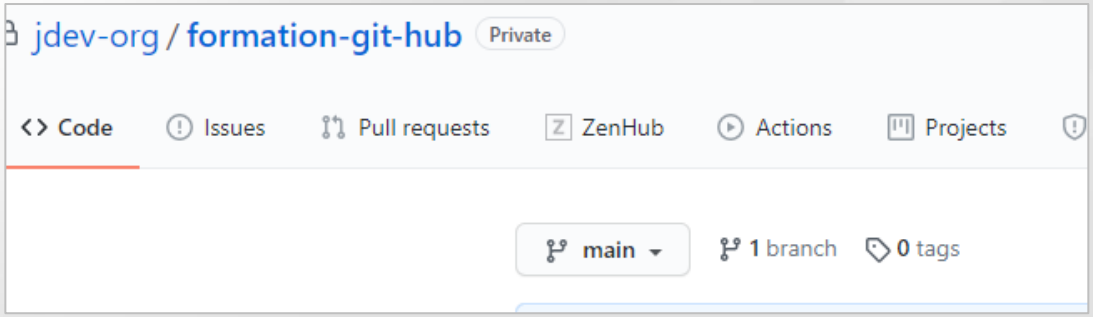


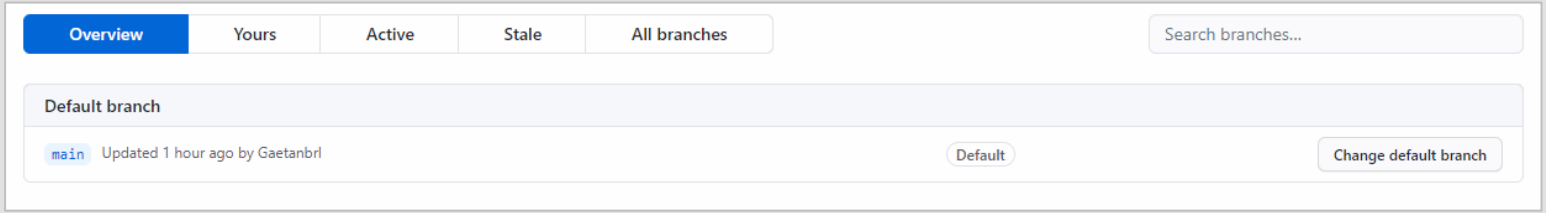
Récupérer en local la suppression d'une branche



Attention
Des doits peuvent être nécessaires
Pour gérer des branches

Github – Changer sa branche immuable





Github – Commit (avec Git)

- ≥ Un commit permet d'indexer une modification
- ≥ Un commit peut être fait en local et ne jamais être envoyé au remote, un push est nécessaire !
- ≥ Un commit doit être accompagné de métadonnées (auteur, date, message, etc…)
- ≥ Une modification peut être faite sur GitHub directement ou dans Git

1. Récupérer les modifications (version propre avant modification)
   ```
   $ git pull
   ```
2. Faire une modification dans un fichier
3. Sauvegarder la modification

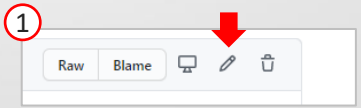
4. Vérifier si le fichier est bien détecté comme non indexé
   ```
   $ git status
   ```

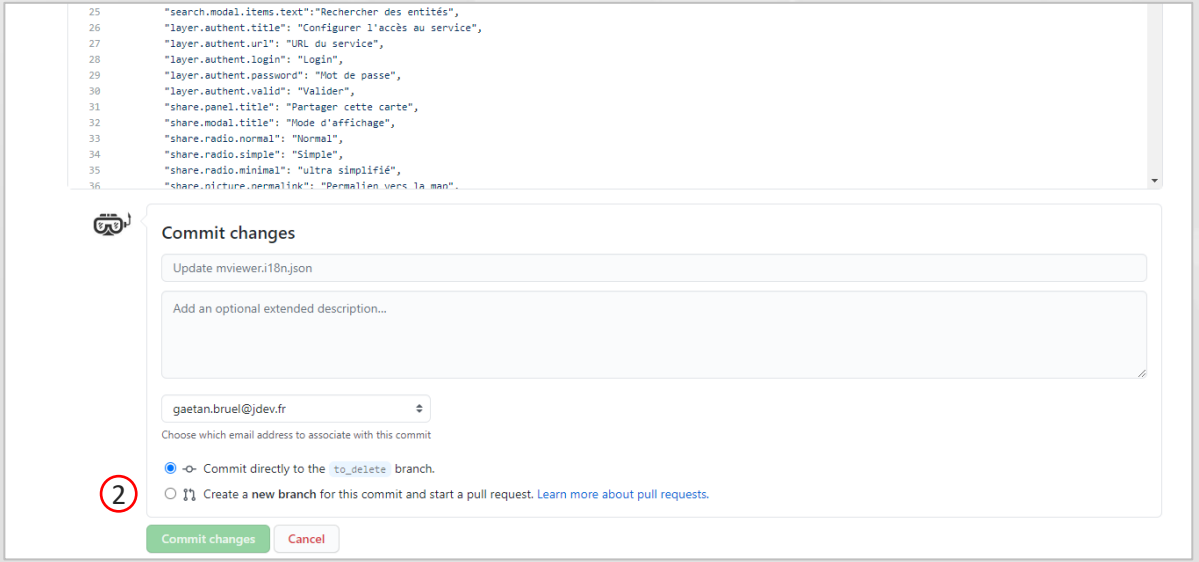
5. Indexer le fichier qui a été modifié
   ```
   $ git add js/mviewer.js
   ```

6. Faire un commit avec un message
   ```
   $ git commit –m "Add new comment line "
   ```
7. Push
   ```
   $ git push
   ```

Github – Commit (avec GitHub)

≥ Il faut être connecté et visualiser un fichier





Attention
Un commit & push est libre sur votre fork mais nécessite une Pull Request pour modifier le dépôt initial (si vous n'avez pas les droits)

Retour sur les fondamentaux

Les bons reflexes

Conseils & bonnes pratiques

- **Commit**
  - Un commit n'est pas sacré
  - Un message est fortement conseillé
  - Toujours faire un pull avant de faire un commit (si plusieurs sur une branche)

- **Branche**
  - Ne pas attendre de finir ses modifications pour réaliser un rebase ➔ Source de conflits !
  - Ne pas hésiter à faire des branches de sauvegarde avant un rebase
  - Si votre état est « divergent » alors que tout est poussé, ne pas hésiter à supprimer la branche en local et la récupérer (fetch + checkout)

- **Merge**
  - Le merge fusionne automatiquement ➔ Source de conflit
  - En cas de soucis avec un rebase « impossible », ne pas hésiter à partir d'une branche à jour et reporter les commits un à un (manuellement ou via un cherry-pick) ➔ Garantir ce que l'on fait !

- **Pull Request**
  - Toujours contrôler le code (de qualité ET fonctionnel)
  - Toujours contrôler les fichiers modifiés
  - Une revue est toujours bienvenue !

Retour sur les fondamentaux

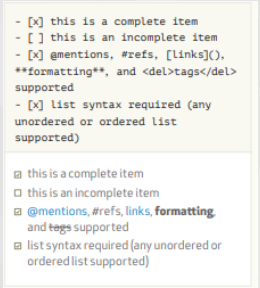
La syntaxe Markdown

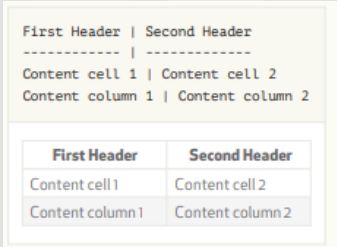
Markdown

Définition :

Markdown est un langage de balisage « simple » créé en 2004 par John Gruber avec l'aide d'Aaron Swartz1,2. Son but est d'offrir une syntaxe facile à lire et à écrire

https://fr.wikipedia.org/wiki/Markdown





- Largement utilisé dans GitHub pour mettre en forme
- Langage largement utilisé
- De nombreux guides en ligne :

https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet
https://guides.github.com/features/mastering-markdown/
https://openclassrooms.com/fr/courses/1304236-redigez-en-markdown
https://wprock.fr/blog/markdown-syntaxe/

Conseil
Mettre en forme c'est aussi bien se faire comprendre !
Une question mal formulée peut aboutir à une mauvaise réponse…

Retour sur les fondamentaux

Documentations utiles

Aide GitHub et Git

≥ Documentation officielles Git & GitHub

https://guides.github.com/
https://docs.github.com/en

≥ Guide de prise en main des commandes Git

https://git-scm.com/book/en/v2
https://guides.github.com/introduction/git-handbook/

≥ Memento Git

http://slam5.lmdsio.fr/lessons/memento-git
https://raphaelhertzog.fr/livre/memento-git/

≥ Forums, collègues…

https://stackoverflow.com/